21 SINGLE BOARD COMPUTERS

www.linuxuser.co.uk

LinuxUser & Developer™

THE ESSENTIAL MAGAZINE FOR THE GNU GENERATION

SPECIAL REPORT

MICROSOFT

How I learned to stop worrying and love Linux

Is Microsoft a **friend or foe?**
It's time for **you** to decide

CEO INTERVIEW

Resin.io

The startup automating Linux containers for IoT

43 PAGES OF GUIDES

- Security: sniff out reliable exploits
- Master GNU Make
- Java: Add AI

PI PROJECTS

Modding Minecraft
Use RDBMS with Python
Guru's guide to GPIO Zero

HUGE ROUNDUP

Get the **perfect** single board computer
21 tiny boards on show

Fedora 27
Reviewed: is this the best GNOME distro in the world?

Raspberry Pi distros
We test four of the best desktop OSes optimised for the RasPi

ALSO INSIDE
» King of Chromebooks: Google Pixelbook
» Install LineageOS

FUTURE

Did you know that European forests, which provide wood for making paper and many other products, have grown by 44,000km^2 over the past 10 years? That's more than 1,500 football pitches every day![†]

Love magazines? You'll love them even more knowing they're made from natural, renewable and recyclable wood

[†]UNFAO, Global Forest Resources Assessment 2005-2015.

Two Sides is a global initiative promoting the responsible use of print and paper which, when sourced from certified or sustainably managed forests, is a uniquely powerful and natural communications medium.

There are some great reasons to #LovePaper
Discover them now,
twosides.info

TWO SIDES

Linux User & Developer
THE MAGAZINE FOR THE GNU GENERATION

Future PLC Quay House, The Ambury, Bath BA1 1UA

Editorial
Editor **Chris Thornett**
chris.thornett@futurenet.com
01202 442244
Designer **Rosie Webber**
Production Editor **Ed Ricketts**
Editor in Chief, Tech **Graham Barlow**
Senior Art Editor **Jo Gulliver**

Contributors
Dan Aldred, Michael Bedford, Joey Bernard, Neil Bothwick, Christian Cawley, John Gowers, Tam Hanna, Toni Castillo Girona, Joe Osborne, Jon Masters, Calvin Robinson, Mayank Sharma, Alexander Smith

Photography
Joseph Branston
All copyrights and trademarks are recognised and respected. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Advertising
Media packs are available on request
Commercial Director **Clare Dove**
clare.dove@futurenet.com
Advertising Director **Richard Hemmings**
richard.hemmings@futurenet.com
☎01225 687615
Account Director **Andrew Tilbury**
andrew.tilbury@futurenet.com
☎01225 687144
Account Director **Crispin Moller**
crispin.moller@futurenet.com
☎01225 687335

International
Linux User & Developer is available for licensing. Contact the International department to discuss partnership opportunities
International Licensing Director **Matt Ellis**
matt.ellis@futurenet.com

Subscriptions
Email enquiries **contact@myfavouritemagazines.co.uk**
UK orderline & enquiries **0888 888 8888**
Overseas order line and enquiries **+44 (0)8888 888888**
Online orders & enquiries **www.myfavouritemagazines.co.uk**
Head of subscriptions **Sharon Todd**

Circulation
Head of Newstrade **Tim Mathers**

Production
Head of Production US & UK **Mark Constance**
Production Project Manager **Clare Scott**
Advertising Production Manager **Joanne Crosby**
Digital Editions Controller **Jason Hudson**
Production Manager **Nola Cokely**

Management
Managing Director **Aaron Asadi**
Editorial Director **Paul Newman**
Art & Design Director **Ross Andrews**
Head of Art & Design **Rodney Dive**
Commercial Finance Director **Dan Jotcham**

Printed by
Wyndeham Peterborough, Storey's Bar Road, Peterborough, Cambridgeshire, PE1 5YS

Distributed by
Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU
www.marketforce.co.uk Tel: 0203 787 9001
ISSN 2041-3270

We are committed to only using magazine paper which is derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this magazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards. The manufacturing paper mill holds full FSC (Forest Stewardship Council) certification and accreditation

All contents © 2018 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have the necessary rights/permissions to supply the material and you automatically grant Future and its licensees a licence to publish your submission in whole or in part in any/all issues and/or editions of publications, in any format published worldwide and on associated websites, social media channels and associated products. Any material you submit is sent at your own risk and, although every care is taken, neither Future nor its employees, agents, subcontractors or licensees shall be liable for loss or damage. We assume all unsolicited material is for publication unless otherwise stated, and reserve the right to edit, amend, adapt all submissions.

FUTURE
Connectors. Creators. Experience Makers.

Future plc is a public company quoted on the London Stock Exchange (symbol: FUTR)
www.futureplc.com

Chief executive **Zillah Byng-Thorne**
Non-executive chairman **Peter Allen**
Chief financial officer **Penny Ladkin-Brand**

Tel +44 (0)1225 442 244

Welcome

to issue 187 of Linux User & Developer

In this issue

» Microsoft loves Linux, p18
» SBC roundup, p58
» Guide to GPIO Zero, p74

Welcome to the UK and North America's favourite Linux and FOSS magazine.

To steal a quote from Mignon Clyburn, a commissioner on the Federal Communications Commission, "a legally-lightweight, consumer-harming, corporate-enabling, destroying-internet freedom order" has passed in the USA. It's sad times, indeed, for net neutrality.

Technically, this order reclassifies broadband as an information service and puts it under the jurisdiction of the Federal Trade Commission. Their job will merely be to make sure ISPs disclose when they do things such as block sites or content they don't like, or throttle services that haven't paid a fee, rather than preventing such practices. However, the fight continues. Just to perk you all up, we thought we'd talk about Microsoft... Sorry about that, but it's about time we discussed Microsoft's change of heart over Linux and open source. It's controversial for many and we'll lay it out for you (p18). This issue, we also take a look at the exotic world of single board computers to see if any pique your interest for your next project (p58). In tutorials, Essential Linux moves on to GNU Make, and we start a two-parter on building an Arduino recorder. As usual, we've packed a lot into the magazine. Enjoy!

Chris Thornett, Editor

Get in touch with the team:
linuxuser@futurenet.com

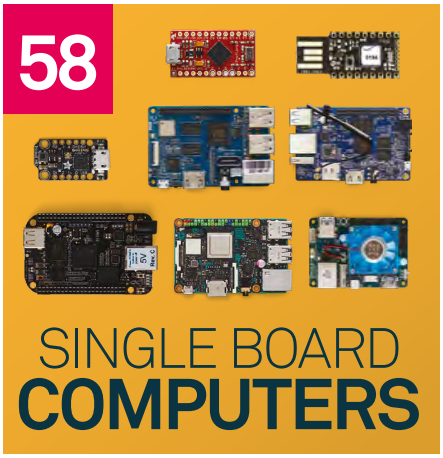
Facebook: facebook.com/LinuxUserUK
Twitter: @linuxusermag
FileSilo **filesilohelp@futurenet.com**

For the best subscription deal head to:

myfavouritemagazines.co.uk/sublud

Save up to 20% on print subs! See page 30 for details

Contents



OpenSource

06 News
More laptop manufacturers opt to disable Intel's Management Engine

10 Letters
Pearls of wisdom before us swine

12 Interview
We talk to resin.io, helping developers deploy code on connected devices

16 Kernel Column
2017 in summary for the Linux kernel

Features

18 Microsoft loves Linux
At least, that's what the software giant says these days – but can you trust it? Is it now time to discard that ancient grudge and forget the past, or should we be wary? We'll lay out the journey that got Microsoft to this apparent new state of harmony, and highlight a few of its open source projects

58 Single board computers
Imitation is supposedly the sincerest form of flattery, in which case the Raspberry Pi Foundation and Arduino AG must be over the moon about the number of single board computers now on the market and within reach of the amateur experimenter. **Mike Bedford** takes a look at just some of the options now available

Tutorials

32 Essential Linux: GNU Make
How to build programs with GNU Make

36 Install LineageOS
Try the fresh, clean mobile OS

40 MQTT: Part 2
How to deploy MQTT on a Raspberry Pi running Android Things

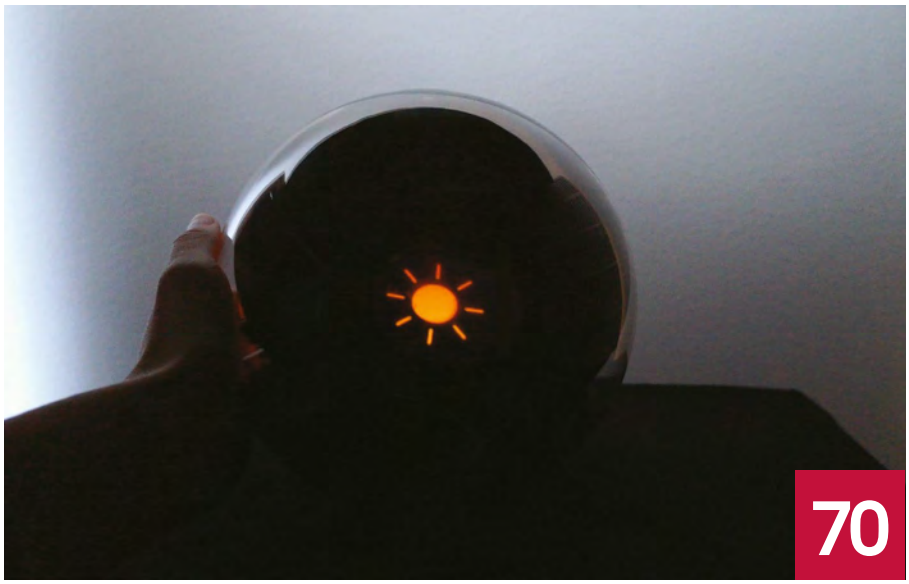
44 Security: reliable exploits
Find and use exploits in pen-testing

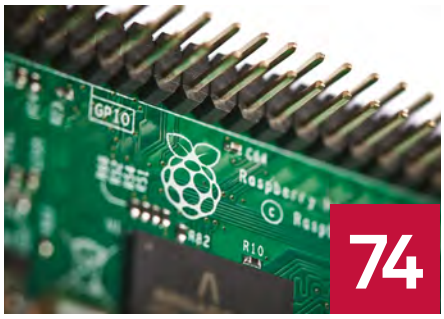
48 Arduino: Build a recorder
Build a Dictaphone-style sound recorder and player using Arduino

52 Java: Advanced concurrency
In the last part of the tutorial, we write an automated bot to play our game

LinuxUser & Developer
Issue 187
January 2018
facebook.com/LinuxUserUK
Twitter: @linuxusermag

FileSilo
94 Free downloads
We've uploaded a host of new free and open source software this month





Practical Pi

70 Pi Project: Sphaera
Taking inspiration from the mystical (and mythical) crystal ball, **Jenny Hanell** and friends create a seemingly magical weather forecasting globe

72 Minecraft & Python
In the final part of our Minecraft Rasperry Pi series, discover how to use Python to mod and tweak Minecraft

74 A guide to GPIO Zero
Make using the GPIO pins easy, fun and expand your interaction with a wide range of components and sensors

78 Pythonista's Razor
Learn how to use a Relational Database Management System with Python to deal with larger and more complex databases

Reviews

81 Group test: Raspberry Pi everyday distros
You can use the little computer as an everyday desktop, with the help of these Linux distributions – but which of the four is best?

86 Google Pixelbook
At £1,000, the Google Chromebook has a lot to live up to. Does it deliver?

88 Fedora 27
Is the first Fedora release since Ubuntu's switch to GNOME still the leading GNOME distribution?

90 Fresh FOSS
Text editor Atom 1.22, Enlightenment 0.22 desktop, OpenCV 3.3.1 programming library and Firefox 57 put to the test

Back page

96 Near-future fiction
Next-day delivery can be pretty important if you're waiting for new eyes

SUBSCRIBE TODAY
Save up to 20% when you subscribe! Turn to page 30 for more information

HARDWARE

Linux hardware retailers spurn Intel

System76, Purism and others eject Intel ME from computers after security warning

Confirmation that Intel's Management Engine (ME) represents an active risk to computers running Linux, Windows and macOS has not only been demonstrated with a proof-of-concept, it has resulted in decisive action from Linux OEMs.

Along with enabling access to various portions of a powered-down system (including network access), it now transpires that flaws in the ME can be exploited to allow arbitrary code execution. Discovered by Positive Technologies researchers Mark Ermolov and Maxim Goryachy, and demonstrated at the Black Hat Europe 2017 event, this exploit bypasses established ME protections. Currently, the exploit can only be used by an attacker with physical access, but this could change. Unfortunately the patching program recently rolled out by Intel does not solve this flaw.

The only way for this vulnerability to be dealt with would be for Intel to stop shipping processors with the ME, and to completely disable it on existing systems.

Already aware of the potential for security breaches in the ME, several Linux device manufacturers have taken steps to disable it. For example, System76 is delivering updated firmware with the ME disabled on 6th, 7th and 8th generation Intel laptops, noting "There is a significant amount of testing and validation necessary before delivering the updated firmware and disabled ME. Disabling the ME will reduce future vulnerabilities and using our new firmware delivery infrastructure means future updates can roll out extremely fast and with a higher percentage of adoption."

> "The only way for this vulnerability to be dealt with would be for Intel to stop shipping processors with the ME"

System76 computers will need to be running Ubuntu 16.04 LTS, Ubuntu 17.04, Ubuntu 17.10, Pop!_OS 17.10, or an Ubuntu derivative, with the System76 driver installed, to receive the ME-disabling firmware.

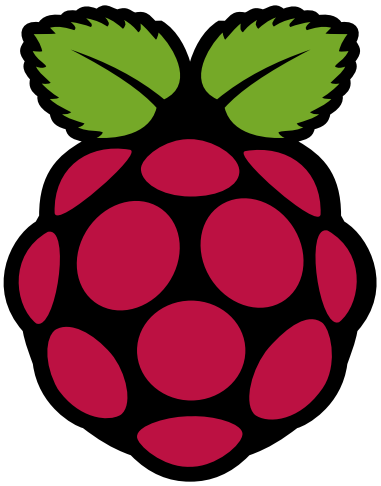
Meanwhile, security-focused OEM Purism has also taken decisive action. Indeed, these steps were taken before Intel's public disclosure in November 2017. In a blog post on 19 October, hardware enableness developer Youness Alaoui wrote "our second generation of laptops (based on the 6th gen Intel Skylake platform) will now come with the Intel Management Engine neutralized and disabled by default." This makes Purism the first manufacturer to disable the ME.

Expanding on this, Purism CEO Todd Weaver told **LU&D** about his company's opposition to the ME. "Intel Management Engine, for over ten years, has been the theoretical worst-case exploit. Purism has fought against the inclusion of the ME in CPUs, from petitioning Intel for an ME-less design in 2016, to reverse-engineering parts of the ME in 2017, to collaborating and cooperating with the other groups cleaning the ME."

Above Many Linux system manufacturers are now disabling Intel's Management Engine

Noting that it is only a matter of time before the Positive Technologies exploit can be conducted remotely, Weaver told us that concerned users can benefit from "Purism's investment in the bundling of secure hardware, TPM, Coreboot, Heads, and FSF-compliant PureOS."

The message is clear: unless and until Intel removes ME, equipment manufacturers will do the job for themselves.



SOFTWARE

Vivaldi browser launches on ARM devices

Hits Raspberry Pi first, with other single-board computers and Android to follow

Vivaldi Technologies, the company helmed by former Opera co-founder Jon Stephenson von Tetzchner, has launched an ARM version of its Vivaldi browser, with the Raspberry Pi its first port of call. While Raspberry Pi 2, 3 and Zero models should run the browser, the original A and B versions of the Pi are likely to encounter significant performance lags.

"Enthusiastic Raspberry Pi users who are looking for a more feature-rich and flexible browser will find Vivaldi a thrilling experience," said von Tetzchner. Beyond the Raspberry Pi, other SBCs (such as the CubieBoard) are being targeted. Not surprisingly, there's an intention to launch on Android, too.

Initially an experimental build, Vivaldi for ARM devices includes most of the features as the x86/x64 version, such as advanced tab management and detailed browsing history.

"Vivaldi is a web surfer's complete toolbox that you can personalise and make your own. We strive to add more flexibility for the thriving culture of computer hobbyists and hope that every owner of Raspberry Pi will have fun using Vivaldi," von Tetzchner added. Speaking to **LU&D**, he explained why his team decided to expand Vivaldi's reach: "It's a geek thing. We are a company that does things that we find fun, and we want to support Linux on various devices."

Vivaldi's features have proven popular. Extension developers for other browsers have been attempting to replicate them – but how long before Google and Mozilla cook those features into the browser? Says von Tetzchner: "Our thinking is [that for] the basic functionality in the browser, you shouldn't require extensions to do basic stuff, and to us, tab handling is a basic feature."

However, regular readers may recall that Vivaldi browser is not open source, which may influence your decision. Vivaldi.com states: "Vivaldi is not made available under one unified open source license. It does contain the Chromium source code with changes made to allow the HTML/CSS/JS-based UI to run. All changes to the Chromium source code are made available under a BSD license and can be read by anyone on **vivaldi.com/source**."

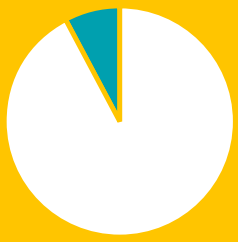
DISTRO FEED

▶ Top 10

(Average hits per day, month to 8 December 2017)

1.	Mint	■	3,218
2.	Debian	▼	1,604
3.	Solus	▲	1,604
4.	Ubuntu	▼	1,534
5.	Manjaro	▼	1,531
6.	Antergos	▼	1,207
7.	Fedora	▼	1,126
8.	elementary	▼	1,006
9.	TrueOS	▼	991
10.	openSUSE	▼	784

▶ This month



A strange month. Mint and Solus aside, there's been a marked decline in hits – and so downloads – of the top 10 distros, but Linux Mint is the clear winner.

▶ Highlights

Lubuntu
Light and fast, Lubuntu is an energy-efficient version of Ubuntu, with the LXDE desktop. With low hardware requirements, Lubuntu is ideal for reviving old PCs, and for virtual machines. This recognised flavour is available as both 32-bit and 64-bit downloads.

Xubuntu
Replacing the usual Ubuntu desktop with Xfce, Xubuntu focuses on "integration, usability and performance". This is another lightweight version of Ubuntu, ideal for low-RAM devices such as old netbooks.

Ubuntu MATE
Offering a more traditional desktop experience to the Unity-era Ubuntu, the MATE desktop environment is based on GNOME 2.

Latest distros available:
filesilo.co.uk

SOFTWARE

CrossOver 17: use Office 2016 in Linux

MS Office users rejoice – the latest version is now Linux-friendly

Despite Microsoft's support for Linux in recent years, there remains a key stumbling block for individuals and businesses planning to switch to an open source operating system: Microsoft Office. While LibreOffice and other FOSS office solutions are adequate, they tend to fall at the final hurdle. Certain features and functions found only in Microsoft Office are missing, which means compatibility in certain scenarios often fails.

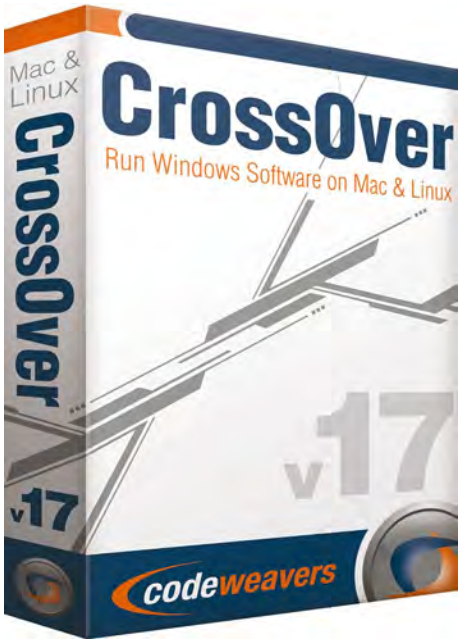
> "While CrossOver relies on Wine, Wine's development also benefits from CrossOver's tweaks"

Microsoft Office 2010 and earlier have been available for Linux for some time now via Wine, but if you're looking for a way to run Office 2016 on Linux, CrossOver 17 has the answer. The commercial version of Wine has added support for Office 2016 (and the earlier 2013 release), enabling business users – and anyone else with a preference for Microsoft Office products such as Word, Excel, PowerPoint and even Access – to enjoy the greatest compatibility between file formats on Linux yet.

While CrossOver relies on Wine, the relationship is symbiotic. Wine's development benefits from CrossOver's tweaks, which include modifications to the source code, user-friendly configuration tools, compatibility patches, installation scripts and technical support. These enhancements (all covered by the LGPL) then feed into the open source version of Wine down the line. Using CrossOver means Wine is supported long-term; packages start from €38.

Microsoft Office 2016 was the most popular software on CrossOver developer CodeWeavers' 'most wanted' list, and joins over 15,000 other supported applications and games. At the time of writing, Office 2016 had most downloads in terms of CrossOver installation, with more than double the number two app (the Steam game client).

Yet another reason to migrate your friends and family to Linux in 2018!



Above CrossOver now eases installation of Microsoft Office 2016 on Linux

LINUX

World's top supercomputers all run Linux

DoD says updates to closed-source software are too slow

Quantum mechanics, weather forecasting, climate research, molecular modelling… the list of uses for supercomputers goes on and on. It should come as no surprise to learn that these systems usually don't run Microsoft Windows.

Over the past few years, versions of Linux have increasingly dominated the list of the world's top 500 supercomputers, and according to **TOP500.org**, every single device now runs Linux. Unsurprisingly, these aren't standard distros; highly customised versions of Linux are used. Because there is no typical supercomputer, the hardware differs with each. Interestingly, however, some do utilise operating systems you'll be familiar with.

- 5 run Ubuntu
- 20 of the supercomputers run RedHat Enterprise Linux (RHEL)
- 109 supercomputers run RedHat's CentOS

Perhaps most interesting is what runs on the fastest supercomputer, Sunway TaihuLight. This Chinese-built device features over 650,000 CPUs, and a combined speed of 93 petaflops, which is equivalent to two million laptops. Also the 16th most energy-efficient supercomputer, Sunway TaihuLight runs a version of Linux called Sunway RaiseOS, with its own customised implementation of OpenACC 2.0.

Rating these computers is a matter of benchmarking, which relies on the LINPACK software library. Supercomputers that make it onto the list must be able to "solve a set of linear equations using floating point arithmetic." TOP500 reportd that Linpack's scalability is its strength in benchmarking supercomputers. "It has allowed us in the past 20 years to benchmark systems that cover a performance range of 12 orders of magnitude… no other realistic application delivers a better efficiency."

It remains to be seen whether Linux's total dominance in the supercomputer arena will last. A Microsoft fightback seems likely…

OPINION

The philosophy of Solus

As the independent Linux desktop OS reaches version 4, **Joshua Strobl** reveals what is driving the distro's development and the new features users can look forward to next

For anyone unfamiliar with Solus, it's a Linux-based operating system built from scratch, balancing the stability of the system with our users' needs for the latest software and a curated rolling release system. Solus focuses solely on home computing and the x86/x64 architecture, which enables us to perform highly targeted optimisations throughout the entire software stack, from the kernel to desktop applications such as Firefox. If you've haven't yet checked it out, here are are some very good reasons why you should.

Solus ships with Budgie, our flagship desktop environment with a modern take on the traditional desktop experience paradigm, as well as a plethora of choice for personalisation to make it your own. Additionally, we make available curated GNOME and MATE editions, with out-of-the-box defaults, such as Dash to Dock for GNOME or our Brisk Menu for MATE.

But Solus is more than just an operating system to us; it's our vehicle for improving the Linux desktop experience and the catalyst that enables and empowers the home-computing user. This vehicle enables us to deliver solutions to real-world problems.

We've developed solutions for simplifying driver detection and management. We've engaged and worked with the open source community on improving the Snap containerisation technology, to enable developers to focus on building their apps rather than the complexities of shipping it to users. We've pioneered solutions for improving the state of Linux gaming with our custom gaming runtime and Linux Steam Integration, available to every Linux user via our Snaps.

Our upcoming Solus 4 release, which should be available when you read this, is the embodiment of our belief that a good computing experience is achieved when integration takes a front seat.

Solus 4 will ship with a further-refined Budgie, with the introduction of window grouping and smarter window switching in its icon tasklist, and our MATE edition will see a visual refresh and improvements to Brisk Menu.

However, the biggest changes are under the bonnet. For starters, we are using Snaps to improve the installation and management of a curated set of third-party applications. The adoption of Snaps enables us to almost entirely eliminate our previous third-party system and empowers our users with the ability to easily install software that may not already be accessible via our existing repository.

Solus 4 is also set to be the first release to include usysconf, our new system configuration interface, which provides a unified approach for configuring the system and safely applying system changes such as user management. usysconf provides a fail-safe binary that's immune to update issues and can also be used as a recovery tool, enabling our users to bring their system back up to full health.

Pairing well with this new system configuration interface, we've developed a 'Quality of Life assistant', aptly named qol-assist. During the lifetime of a rolling-release Linux operating system, such as Solus, new problems can occur that are often complex to deal with. For example, migrating active users to user groups to enable new features or functionality, or groups required by udev rules, can be particularly troublesome and

> "Solus is more than just an OS to us; it's our vehicle for improving the Linux desktop experience"

may require manual intervention. qol-assist eliminates the need for manual intervention by automatically handling migrations when new software is installed, or whenever we update the assistant. Existing Solus users have, in fact, already experienced qol-assist, when we automatically migrated administrative users to new fuse, plugdev, scanner and user groups.

In essence, combining the capabilities of usysconf and qol-assist not only enables us to improve the experience automatically over time, but gives our users the means to keep their system rolling.

So if we've piqued your interest, you can download Solus today or even keeping track of development at **https://solus-project.com**. Give it a go and let us know what you think – your feedback matters.

Joshua Strobl is the communications manager for the Solus project, not to mention a Go programmer and web developer

COMMENT

Your letters

Questions and opinions about the mag, Linux and open source

Feature
Linux in Space
LINUX IN
SPACE
Linux is key to much of NASA's space exploration but, paradoxically, its influence isn't universal, as Mike Bedford reports
NASA's Advanced Supercomputing division (NAS) has not one but four supercomputers. Top spot goes to Pleiades

Above We agree that the magazine needs to be "serious and balanced", but that doesn't mean we can't use a splash of colour to complement what we're writing about

Black and white

Dear LU&D, First, thank you for a great magazine. So far, in my opinion, the balance of articles has been better than ever on your watch, particularly the security stuff. It goes without saying that you cannot please everyone!

Can I make a critical observation? I am in my sixties, have been involved with computing since the ZX Spectrum, and continue to be interested. However, although the content is great, the look and feel of the average page is pretty dreadful, with deep inverse colours behind trendy thin fonts – more like a 1970s teenage magazine than a technical magazine, and quite difficult to read.

You have an opportunity to create a 'style' (such as on your tutorial pages) across all content, that will mark you out as serious and balanced. Black print on white paper may be old-school, but everything else looks like a throw-away fashion rag.

Tim Morris

Chris: Thanks for your kind words regarding the magazine's content, Tim. In terms of security articles, we've been helped greatly by having Toni Castello Girona as our enthusiastic and knowledgeable security writer. As usual, we'd love to hear what topics interest readers, as we have a few ideas for features this year and plan to cover some of the key events, such as InfoSec.

I'm not sure I'd agree with describing our layout style as a "throw-away fashion rag". I've been very fortunate to have a very talented art person, Rosie Webber, working on the magazine since I became editor. We've changed some aspects of the magazine, but it's mostly been a slow evolutionary process rather than a Cambrian event as we've only been able to refine and refresh pages between issues. Rosie is doing a sterling job and we will continue to look at ways to improve each issue. I'm not averse to a splash of colour even in technical magazines, but I agree that "serious and balanced" is probably the right editorial approach for us, though I don't think it means we need to be dull and boring at the same time.

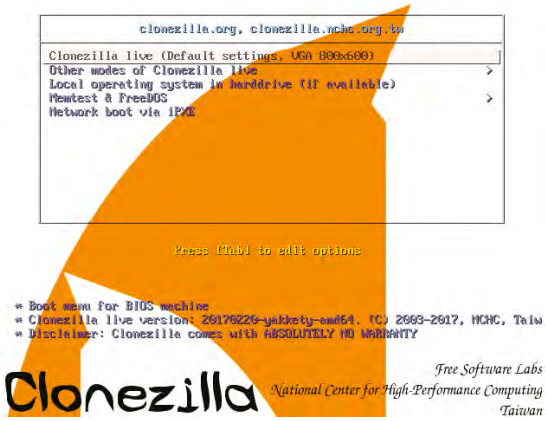
I do take your point about inverted colours and using thin fonts. We have increased the font weight when we've had a background colour, but maybe we need to look carefully at its use. I must concede that experience has taught me that old hands that have spent decades staring at flickering terminals can find coloured boxes tricky to read.

Backup buddy

Dear LU&D, I'm enjoying the magazine but wondered if you could help? I'm looking for something to backup Red Hat Linux boxes. At the moment, I'm using Acronis True

GET IN TOUCH!

Got something to tell us or a burning question you need answered? Email us on **linuxuser@futurenet.com**


clonezilla.org, clonezilla.nchc.org.tw
Clonezilla live (Default settings, VGA 806x600)
Other modes of Clonezilla live
Local operating system in harddrive (if available)
Memtest & FreeDOS
Network boot via iPXE
Press [Tab] to edit options
* Boot menu for BIOS machine
* Clonezilla live version: 20170220-yakkety-amd64, (C) 2003-2017, NCHC, Taiw
* Disclaimer: Clonezilla comes with ABSOLUTELY NO WARRANTY
Clonezilla
Free Software Labs
National Center for High-Performance Computing
Taiwan

Above Clonezilla is a Linux favourite for disk cloning, disk imaging and data recovery

Image for backing up the Windows machines we have here, but is there something equivalent to that for Linux that I can use?
Jacob Combs

Chris: There are lots of backup tools out there, but it's probably a choice between rsync and Clonezilla. In fact, if you look at other backup tools, you'll find that many are front-ends for the rsync utility (Back in Time, for example). I've not used Acronis personally but I would say that Clonezilla is likely to give you the same features even if it probably doesn't offer an attractive interface while doing the job.

The choice depends on what you want to backup and restore. If you need something for drives and partitions then try Clonezilla. Rsync is better for files and directories. It's important to remember that rsync just copies the contents of one directory to another and that can be anything from a file to an entire filesystem, but it won't create or format filesystems, so don't expect it to backup or restore the boot sector, partition structure of a drive or the format of a partition. So use Clonezilla for recovering from hard drive failures and rsync for recovery of the contents of a drive.

Bleak Friday

Dear LU&D, Thanks for a great read, I've been a subscriber for a few years now and it's only getting better and better. I do have one request: please dial down the number of subscription advertisements! Black Friday (or should it be Black December? The whole thing seems to go on all month now) was particularly bad with flashing GIFs and 'last minute' deals. I like a bargain but it got very silly at some points.
Sanzida Ahmed

Chris: Sorry about that. Yes, the company has been rather enthusiastic this year. My own feed on Twitter was an explosion of gaudy flashing widgets for a while, as everyone was trying to sell me something at a bargain-limited-reduced-exclusive-must-end-soon price. Strangely, I even encountered a number of companies sending out 'must-read Black Friday'-type emails about how nice they were for not doing Black Friday!

I have to acknowledge that the deals were quite brilliant, though. At one point we were up to 42% and 43% off EU and US print subscriptions, plus an extra 20% on top of that. That's not exactly the easiest of deals to understand in a microsecond, but I can't really complain as, for a change, our American friends were getting a great offer.

As it stands, Linux User and Developer smashed its subscription targets and is currently top in the company charts. That may not mean much to you, but to us that means we must be doing something right. In this crazy and volatile world of magazine publishing, we'd much prefer to focus on the subscribers who join us on our journey, and what they want from us, than worry about what will pull in shelf-browsers for a one-time purchase. In fact in many ways, we feel they are very different readers of the magazine.

CORRECTION!
Apologies to Stefan Vorkoetter – whose beautifully designed Raspberry Pi tablet we covered in **LU&D** 182 for Pi Projects – for incorrectly saying that he attended the University of Washington. In fact, he attended the University of Waterloo in Ontario, Canada, which is roughly 2,470 miles away on the other side of the continent.

Below You weren't fans of the torrent of adverts for Xmas and Black Friday. We're so sorry that we're going to make a flashing GIF to apologise

INTERVIEW RESIN.IO

Pain-free Linux containers for IoT

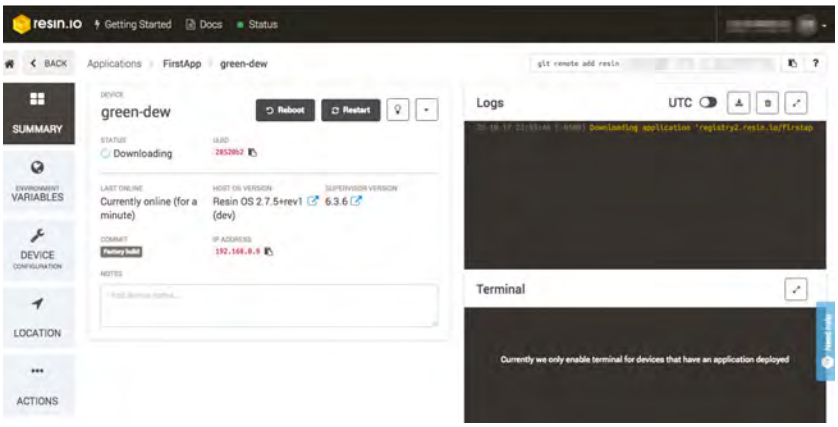
We interview resin.io, the company behind the extremely popularly SD burning app, Etcher, and the resinOS software platform that helps developers build, deploy and manage code running on connected devices

Alexandros Marinos is the Founder and CEO of resin.io, a Seattle-based startup that tries to make deploying and updating Linux containers on connected IoT devices as easy as possible. The company's buzzline is 'We make IoT work'.

Could you give us an overview of what resin.io does as a company, for readers who haven't encountered you before?

resin.io is about the person we call a fleet owner. They have a lot of devices and mobile computers – either industrial or something as simple as a Raspberry Pi. They're out there in the field, deployed for doing anything from a digital signage screen to a smart-building monitor, a smart meter or even something as crazy as a drone. We have a customer that does underwater turbines off the coast of Australia. It could be literally anything – so long as it's a Linux computer that's sitting out in the field and needs to be managed.

So at resin.io, essentially we take data centre technology – the same technology we use to manage and update servers in a data centre – [and use it] in a context where there is a lot less processing power, a lot less stability in terms of actual electrical power. Everything is worse, basically, outside. You don't even control the environment. Obviously it could be hot or cold or anything. So we try to make data centre technology work outside its familiar context, essentially. People can build these fleets of devices and manage them in a way that a developer would find civilised rather than having to do things that are incredibly painful, slow and error prone... Or the alternative is to do nothing.



Above The resin.io dashboard is an API client for viewing detailed information about your devices

Is it true that you got into containers working in rubbish collection?

Oh yeah, there's a funny story there. In London there used to be bins with recycling units with screens on them, and basically, we were the team that was running the technology for that company. And that's kind of how we started, with a set of questions around: "Oh, well, these machines are super-powerful. They look very much like servers, and we want to manage them out there." But we just came across all sorts of problems where we would have to go around with keyboards and USB sticks and drills to open these things up and reflash them. And we'd have to walk around a hundred separate spots where they had these units, just to make sure everything was fine.

I kind of intuitively said, these tools must exist. It can't be that if I'm working in a cloud, everything is perfect, everything is automated, and then the moment I leave those walls, I'm back in the 1980s. It didn't make intuitive sense to me. So we looked and looked and it turns out no, there was nothing there. So we said, "Okay then. That's something we can fix."

At that time, Linux containers and Docker were just starting. We saw that Linux containers were going to be the loophole that was going to allow us to push a lot of the cloud workflow and technologies through to the embedded device. Early on, everybody thought that that was a crazy thing to say. They didn't quite understand even how Linux and embedded computers would intersect, never mind how Linux containers come into the picture.

Now all of that is a lot more acceptable and accepted, and we get a lot of credit for having established that kind of foothold very early on. But back then, I was like the wandering mad man in Silicon Valley, trying to convince people that this was the future.

How do you deal with the security side of things?

To begin with, there is an essential security element to resin.io itself. You can't secure what you can't update, right? If you do not have a way to update your system, then any flaw that comes out in the Linux kernel, whether it's Heartbleed, a Krack, or

Meet the Beast

Resin.io has a company 'pet' called the Beast. This is an in-house cluster platform for testing resin.io product updates. The latest incarnation, version 3, has 336 Raspberry Pis and is designed as modular 'computational LEGO' so it can be expanded to build large clusters.

Since resin.io's focus is infrastructure, the Beast came out of a need to visualise what the company did in a clever way while talking to people for five minutes at a conference: "We'd say, 'Oh, you know, you can have all these devices, and instead of one single command, we build your software and deploy it,' says Marinos. "And that's very fine to say, and people will nod along. But it's abstract. You have to imagine all these devices, and you have to imagine what happens."

In an early demo, you could touch one screen and see the touch ripple out to the other screens around it: "This is the kind of demo we like to do," says Marinos. "There's some kind of interaction, and there's some kind of swarm behaviour from the devices."

Initially, resin.io started building little clusters. "The first one, I think, was just three devices that I had in a suitcase that I would go around to VCs and offices and pitch. And then we'd start adding more and more. I think sometime in 2014, we had a 120-device cluster," says Marinos.

He says the first Beast took 10 days to build with just him and Shaun Mulligan: "Of course, the first one was really literally a beast. It was extremely unwieldy to transport anywhere."

So they decided to use a modular setup: "That's why I'm saying I don't know how many we're going to build, because we essentially built these 12-device tiles, and they're small Beasts, essentially. They can be attached to each other, and then they get network and power from each other."

It's a clever setup which Marinos and his team are proud of, although he admits it took an unbelievable amount of time and effort to hit all the constraints at the same time. "Because when you're building something like that, you have to think about heat, power, networking and structural integrity, and, of course, you have to think about the software and the hardware."

resin.io wants the Beast to be an open design, and as it's modular anyone can build their own networks, not just from Raspberry Pis. Their project explanation mentions Beaglebones, Odroids and Orange Pis, and indicates you can use any single board computer you want. For more details head to **https://hackaday.io/project/27636-the-beast**.

Marinos adds: "It's easy to say 'We're going to connect them all together and we're going to mine Minero or something,' but to me, that's not satisfying because it doesn't really get to the core of what these devices are supposed to be doing: sitting somewhere and interacting with the world."

When we spoke to Marinos he was hopeful resin.io would be able to set up the complete Beast for the first time at the Maker Faire Rome in December. And, judging from images and GIFs posted on the company's Twitter account (**@resin_io**), it certainly made a good impression.

Above Resin.io's pet 'cluster monster' of 360 Pis was on show at the Maker Faire Rome

anything else, you're done. It's there, and you have no way of fixing it. So having the ability to update your devices means you have the ability to secure your devices. So there's an essential security element to what we do. Now, beyond that, of course, if the update mechanism itself is compromised, you have another problem. It's essential that we also secure our own systems.

We do several things – a lot of it is just making sure we do the standard practice properly. People know how to secure cloud servers. We have two-factor authentication; we encrypt all the traffic to the devices in the cloud and all of those things. Then the devices themselves have unique API keys that can be revoked.

So let's say one of them gets compromised or it gets stolen... you can immediately revoke it. The system will give it no more information. But we also compartmentalise a lot. So the device only ever has access to information that pertains to that device. Even if one of our servers or one of our micro-services is compromised... internally we have a lot of compartmentalisation.

Over time, we're starting to work on things like working with the on-chip security on these devices to have end-to-end assigned payloads and things like that. But we're following the hardware as it develops and as it gets released in the world. Because right now, most devices do not have these capabilities. So even if theoretically our operating system could support them, most people wouldn't be able to use them anyway. So that's something that's next for us.

You've been quoted saying "It's not a trivial thing to build a decent open source project." I was curious to get some idea of your experience of the non-trivial side of open source projects.

To qualify as an open source project, all you really have to do is go to GitHub and put some source

> "I was like the wandering mad man in Silicon Valley, trying to convince people that this was the future"

Above Balena uses the Moby Project to modify the Docker code base to, for instance, reduce its size to 27MB and make sure that it writes to the disk every single time

code up, right? And theoretically you're an open source project. Our most successful open source project to date has been Etcher, which is an SD-card writing program that is now being recommended as the default – the Raspberry Pi, the BeagleBone, Bluetooth 4.0 USB sticks, it's recommended.

So it got to a place where it has thousands of users every day. That's a free and open source project that we started. But a lot of what that team did was not just about putting out the code. They spent a lot of time chatting with users, dealing with user issues, documenting their codebase so that people could just jump in, enabling others to really engage with the substance of an open source community – not just putting some source code out there and claiming a victory.

You have to engage with the community on a level where they feel you're not just going through the motions, and that you're actually serious about staying around and supporting the project over time. Then the community will put in the time to help work with you and highlight issues.

I mean, Etcher has become so successful because we'll have an issue, and we'll ask them to come in and debug. Because there's all these different system configurations that you'll find; all these different Linuxes; all these different drives; and all these things that can go wrong while you're writing an SD card. It's actually a surprisingly complicated task. But we improve by just working with users.

What open source projects does resin.io contribute to?

We have several. If you go into our repository, there's literally hundreds of projects. Some of them have a lot of success, others are more tactical. They do their job, and some people will like them and use them, and everybody else just uses them through the product.

resinOS is one where we're putting a lot of work, so our operating system is open source. There are people like Home Assistant, for instance... There's a team that's built this home automation server, and they recently put out their image, which is based on resinOS. So they took resinOS, they added their own software on top, and released it out there.

resinOS is kind of like a CoreOS for the edge, for the embedded device. So it's very simple, very stripped-down. It's a very good base for other software users to use as a foundation. But overall, we are actually moving towards open sourcing significant parts of resin.io itself. So the ability to essentially create an open source management server for a fleet that is not hosted by resin.io is our goal. The agent, the operating system, some parts of the back-end – all those things are open sourced now, but they will gain a lot more meaning when we pull them together into a coherent product that people can actually run.

Tell us about Balena as well – what does that bring to the table?

The origins of Etcher

Born out of the embarrassing number of steps involved in burning a humble SD card for resin.io's new users, the company built its own SD card-burning app and open sourced it. Alexandros Marinos says the team didn't have an expectation that Etcher would become so popular: "It definitely started small, but I think in retrospect, the time we put into the project, I think, was one that gave it a good shot as a success." Essentially, Marinos says, resin.io was trying to solve its own problem and in doing so solved other users' problems. Some of those users were publishers themselves, which Marinos says created a "baked-in viral loop".

"People would have all sorts of problems, such as the SD card wasn't big enough, the download was incomplete, the SD card was locked, or the dreaded and worst of all cases, the SD card was corrupted. Because makers – you know, they'll just use the same card over and over again. It may have a bad sector or something. How would they know they have that, right? But when you have some corruption on your card, that means our software has some random place that is broken. The error, if we do see it, it'll be random. We won't be able to debug it."

In a sense, resin.io felt that had to intervene in the process at the point where you write your card. "So we kind of said, 'Okay, we'll do that. We'll do it cross-platform, we'll do it open source, so that everybody can trust the software, everybody can use it." Like all of resin.io's software Etcher focused on ease of use and has made a process that is actually quite complex very easy to use. Etcher is such a success story, the company decided to develop hardware to complement the burning software; it will soon release duplicator hardware that comes with Etcher installed, and enable you to duplicate anywhere from one to 16 SD cards or flash drives, or write a disk image to 16 devices.

Above Etcher, the popular open source SD card-burning app developed by resin.io, is now in the top 1,000 projects on GitHub

As I mentioned before, we were the first sort of people to seriously do containers in IoT. We were actually the first people to port the Docker engine to the ARM platform altogether. When we started, that wasn't a thing. So we had to do it.

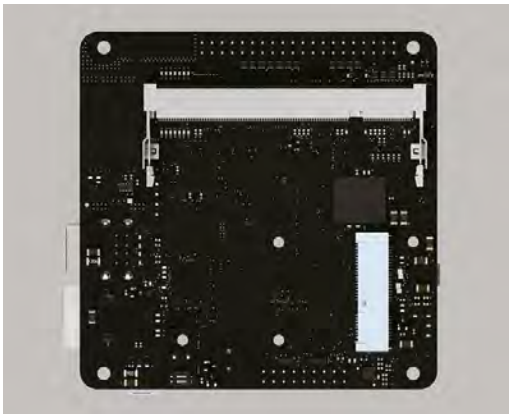
Over time, as we've been working with Docker, we're building resinOS, which is kind of around Docker, and seeing how these nodes behave in production. We've been fortunate enough that people have put thousands of devices out there with resinOS, and we see what happens.

As we've built more and more versions of resinOS with Docker inside it, we also started building a set of patches or modifications or workarounds for some things in Docker itself, to make it function like it should in an embedded context. Docker put out the Moby Project – a toolkit they used to make Docker, with the explicit intent of allowing other people to make Docker-like engines for different contexts. It's almost like they made our wish come true.

So we basically took the Moby Project, which is the Docker code base, and we were able to modify it, remove some things that we didn't feel were necessary for the context that we were in. One of our bigger problems was that Docker was growing in size. When we started working with it, it was 25MB; now it's 100MB.

We have a fixed size on the device that we can allocate to the operating system. We can't tell our user, "Can you reduce the amount of data you've got on this device because Docker got bigger?" So with Balena we've gone back to being about 27MB, because we just ripped out a lot of different things, and we compile the codebase in a different way. But we also have added a lot of robustness. Characteristics like, for instance [...] Docker will not behave well if you just pull the power while it's downloading a container. That's fine, because it's meant to be used in a data centre, but for our contexts, the devices get their power pulled all the time. You can't imagine the sorts of things that happen to these devices out there. So we have to optimise it differently. Balena is more conservative; it's a little bit slower, but it'll make sure that it'll write to the disk every single time. That's a trade-off that we want to make and Docker doesn't want to make.

I think the headline feature for Balena has been container deltas. Even with the layer feature of Docker, you're still liable to have to download hundreds of megabytes per update, especially if you change the base image of a container or even if you add a new package. With Balena, container deltas look at the two versions of the code of the container – what's on the device, and what we want to get on the device. We compare them, and we find the absolute difference between the two things, and we send it to a device. Then the device reconstructs the container we want to put there, based on the one it already has, and the difference that we sent it. That can result in things as crazy as 10-70 times smaller updates. It means that you can go from something that would have been 200MB and make it a 2MB or 3MB update. Imagine having thousands of devices in the field and paying 3G bandwidth for that – [our system provides] thousands of dollars or maybe more of saving for updates.



Above A sneak peek at resin.io's baseboard for the Pi, which will have industrial power input (5V-24V), a cellular modem or a LoRa mode, access to 5GHz Wi-Fi and onboard storage

I just want to make clear, we've also released Balena as an open source project. You don't have to use resion.io or resinOS to use Balena. You just go to **www.balena.io**; you download it; it'll work like Docker. All the usual Docker commands will function. It's a drop-in replacement.

Is there IoT hardware in resin.io's future?

Yes... We have seen a very specific pattern where a lot of users take the Raspberry Pi and they try to go very, very far with it, which is to the credit of the Raspberry Pi Foundation. But the Pi itself, if you ask the Foundation, is intended for education. So they have recently put out the Compute Module, which is essentially a Raspberry Pi without all of the hardware around it. It's just the CPU and a couple of other components.

So what we've done at resin.io – it's a very good segue from Balena, because we're pretty much doing the same thing, just in hardware. We saw what our users need from the Raspberry Pi, and we put all these observations together into creating a baseboard for the Raspberry Pi. So you're still buying a Compute Module. You're still funding the Foundation and its mission. But you're getting a board that's optimised and created in the context that you're trying to use it.

Our users love the Pi. They don't want to move away from it. But they do need this extra level of robustness for situations where [a] device is sitting there unattended for months or years. We already have prototypes, the plans are quite far along.

> "We've been fortunate enough that people have put thousands of devices out there with resinOS, and we see what happens"

OPINION

The kernel column

Jon Masters reflects on the year that was 2017 – moving from Linux 4.10 to Linux 4.14, and the ongoing development towards what will become 4.15

Jon Masters
is a Linux-kernel hacker who has been working on Linux for more than 22 years, since he first attended university at the age of 13. Jon lives in Cambridge, Massachusetts, and works for a large enterprise Linux vendor, where he is driving the creation of standards for energy-efficient ARM-powered servers.

Linus Torvalds has announced Linux 4.15-rc3, noting that it was "big even by rc3 standards. Not good". RC3s are often large, due to maintainers posting fixes for issues discovered in the early part of a development cycle, so it isn't too surprising, but Linus nonetheless hopes that things would calm down over the coming weeks. We covered some of the new features from 4.15 in the last issue. One of those topics that we didn't quite have room to squeeze in was the addition of thousands of SPDX identifiers to the latest kernel. These human-readable tags, which will grow over time with follow-on contributions, allow automated software to determine the licence of individual source files and better track compliance with open source licences – important for commercial developers.

A 2017 retrospective

As in previous years, 2017 featured new developments in many different areas of the kernel. As has been generally true in recent years, many of these were loosely woven together through a common thread of 'optimization'. Specific areas of kernel enhancement included the usual heavy dose of virtual memory (memory management) subsystem reworks. An example of this was support for background writeback, which aims to finally fix the sluggishness that can affect Linux laptops following the insertion of a removable storage (or slower) device. In this situation, writes to the USB can effectively block regular I/O activity on other system disks. Another effort to improve VM efficiency came in the form of VMA swap read-ahead, which changed how Linux manages its swap space to better handle bringing its contents back into memory once it is needed again. Instead of preemptively bringing in often unrelated physical memory pages that happen to be stored close together on the swap, Linux is now optimized to bring in pages that are closer together in virtual memory, which is more useful to real programs.

Linux has long featured an approximately two month development cycle of a 'merge window' for new features, followed by seven or eight weekly release candidates prior to each new release. For many years now, a few ancilliary kernels have been produced and maintained by well-known members of the kernel community. These include development trees, integration trees (such as linux-next), and the 'stable' kernel series from Greg Kroah-Hartman. These kernels provide back-ports of certain fixes to older releases used by distros and projects, typically living for one development cycle. Once in a while, a stable kernel is blessed as being a supported LTS (Long Term), meaning that it will get updates for a few years. In September, Greg announced that he would be doing a six-year LTS kernel. This extraordinarily long period will serve to benefit projects such as Android, where a given release typically stays on the same kernel base throughout its lifetime.

New kernel enhancements over the past year that enable microprocessor features included those targeting both high-end server systems, as well as consumer-grade laptop and mobile hardware. On the server end, Intel's CAT (Cache Allocation Technology) came in through Linux 4.10 back in January. CAT allows the LLC (Last Level Cache, otherwise sometimes known as the L3) to be partitioned up into slices that are assigned into specific virtual machines and/or containers. This results in more finely grained control over the performance of, especially, cloud-computing resources, where the impact of cross-VM or cross-container cache interactions between unrelated workloads needs to be minimized. In a related effort, AMD introduced SME (Secure Memory Encryption) in Linux 4.14. This will combine nicely with another AMD feature, known as SEV (Secure Encrypted Virtualization), to allow virtual machines and containers the ability to isolate their contents from prying eyes; even the 'trusted' sysadmin can be prevented from spying on workloads. This is sometimes referred to as the Snowden Defence, but it will also allow novel new opportunities to run workloads in less trusted environments (and countries).

Linux finally gained support for USB-C devices with the release of kernel 4.12 in July. Using the new tcpm (Type-C Port Manager), users will soon

be able to plug USB-C devices into systems running their favourite distro and have it do the right thing. That might include Power Distribution being used to supply power to a mobile phone from a laptop, or vice versa. The 4.14 kernel release in November added support for five-level paging on x86 machines. This will allow for greater than the previous 48-bit (35TB) upper limit imposed by Intel's traditional 'canonical addressing' and extend support for physical memory up to 56-bits (9PB). This means some truly crazy amounts of RAM could be installed into future servers (or even laptops), but is really aimed at the rapidly developing world of non-volatile memory devices that are treated like memory by the operating system, and mapped into the physical address space just like other memory devices.

On the non-x86 architecture front, there were many developments, including support for SVE (scalable vector extension – think AVX done right) on future ARM devices, but by far the most interesting of these will likely turn out to be the introduction of RISC-V support into the upstream kernel. We've mentioned RISC-V in a number of issues over the past year because of its application of open source principles to the development of hardware. Over the coming year or two, new development boards will be introduced containing completely open source chips that are capable of running Linux. This can only serve to further help democratise computing and make many of the more obscure aspects of system design accessible to those entering this space.

> "Some truly crazy amounts of RAM could be installed into future servers (or even laptops)"

Linux turned 26 years old in August, and at this point is a mature project with over 1,000 developers regularly contributing upstream, and many more spread throughout companies across the globe. While there are occasional controversies to handle, generally speaking the state of the Linux union is strong and only grows over time. I can't wait to see what 2018 has in store for us all, and of course **LU&D** will be covering each new development.

What the future holds

Wu Hao posted version 3 of a patch series entitled Intel FPGA Device Drivers. The patches "provide interfaces for userspace applications to configure, enumerate, open, and access FPGA accelerators on platforms equipped with Intel(R) PCIe-based FPGA solutions and enables system level management functions such as FPGA partial reconfiguration, power management and virtualisation". These patches lay the groundwork for mainline Linux kernel support for FPGA accelerator solutions that are likely to come to market over the next few years. Companies such as Amazon are already making FPGAs (field programmable gate arrays) and other 'reconfigurable' logic (essentially, programmable hardware) devices available in novel cloud-based offerings. These allow developers to design versatile custom hardware acceleration (that is, offload) devices that assist the CPU in doing its work.

Since FPGAs can be far more energy-efficient at much higher performance than doing work in pure software alone, the incentive exists for their adoption in emerging workloads – especially if they 'just work' with upstream Linux.

Finally, this month Wei Hu posted some fixes for an RDMA driver that enables certain Huawei networking features on the company's chips. Ordinarily, this wouldn't in itself get a mention, but it sparked an interesting side conversation into what the default target for bug fixes should be: the latest in-progress RC kernel, or 'linux-next', targeting the next development cycle. The general consensus seemed to be with Leon Romanovsky, who said: "If you treat all unmarked patches (without mentioning in cover letter or subject title) as targeted to for-next, it will make your life much easier than trying to pick each patch alone".

MICRO

How I learned to stop worrying and love Linux

> "And finally, what I think you will see is the intelligent, closed organizations moving increasingly in the open direction. **So it's not going to be a contest between two camps, but, in between them, you'll find all sorts of interesting places that people will occupy.** New organisational models coming about, mixing closed and open in tricky ways. It won't be so clear-cut; **it won't be Microsoft versus Linux – there'll be all sorts of things in between.** And those organisational models, it turns out, are incredibly powerful, and the people who can understand them will be very, very successful."
>
> - British author **Charles Leadbeater** speaking on The Era of Open Innovation at TED 2005

S♥FT!

Does Microsoft love Linux? Should you believe Satya Nadella, an appealing and less aggressive, chair-throwing CEO of Microsoft, when he says something like that? Is it now time to discard that ancient grudge and forget the past, or should we be wary? Whether or not you agree with the quote we've splashed across the bottom of the left-hand page, it does seem incredibly accurate of developments at Microsoft, particularly in the last five years.

The mix of closed and open software "in tricky ways" is certainly likely to resonate for any Linux user who sees free software as a philosophical movement first, and who has aspirations to liberate the individual user while flatly refusing to countenance the use of proprietary software. For others, Microsoft's adoption of open source software development, its focus on interoperability and greater transparency may be gratifying, particularly if it's made your work life easier. We'll take a look at that in this feature: the history of Linux and free and open source software (FOSS), and how events have enabled Microsoft to take this more open route.

Given that most mentions of Windows in Linux magazines generally involve how to escape it, it's hardly surprising that asking **LU&D** readers what they thought of Microsoft's move to open source produced answers mostly concentrated on the operating system. Until recently, the culture at Microsoft was also Windows-centric. In fact, delaying or removing new features from other products that might damage Microsoft Windows was a recognised strategy within the company.

Soon after Nadella was appointed as CEO, he announced his Cloud First, Mobile First strategy and promptly dethroned Windows. While both Microsoft Windows and Office continue to make a large proportion of the company's profits, Nadella sees the future prosperity of the company as being in the Azure Cloud platform.

By completing the work started by his predecessor Steve Ballmer, Nadella aims to grab a healthy slice of the cloud business servicing web-based applications and supporting growing markets, such as AI. As Nadella famously said in a Wired interview soon after his appointment: "If you don't jump on the new, you don't survive." And thanks, in no small part, to the success of Linux and open source in the server and enterprise markets, Azure has to become both the best global cloud platform and the best home for developers. It can't do that on its own anymore; it has become an established business staple that for a platform to be pervasive in today's world, it has to be open source.

The grand plan is to keep spending. Currently, Azure has over 100 data centres, but capital expenditure on new centres is set to double to $9 billion a year. Microsoft has also accelerated its contributions to open source projects. "We want to reduce friction," says Julia Liuson, Corporate Vice President of the Developer Division. "So they [developers] can quickly build and deploy open source-based solutions without having to maintain the underlying servers and operating system." According to Liuson, Microsoft has over 16,000 contributors on GitHub, and has released more than 3,000 open source projects. Luison also says that "Microsoft's open source programs office tracks nearly 10,000 open source components across the company."

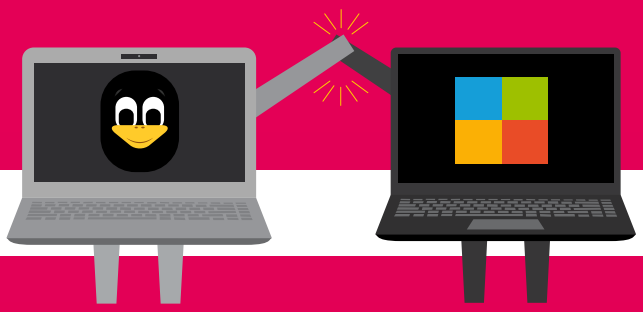
> "Microsoft has over 16,000 contributors on GitHub and released more than 3,000 open source projects"

Whether you can reciprocate Microsoft's affections is for you to decide, but let's lay out the journey that got Microsoft here and highlight a few of its open source projects.

AT A GLANCE

Where to find what you're looking for

- How Microsoft learned to love Linux **p20**
  A chronological look at the key points in Linux history that paved the way for Microsoft to change its mind about Linux and open source
- Microsoft's adventures in open source, **p24**
  What was the first product that Microsoft ever open-sourced, and what does Microsoft have to offer open source developers?
- An interview with Microsoft's Martin Woodward, **p27**
  Meet the engineer that migrated 65,000 Microsoft engineers to Git, and learn what it was like to be part of Microsoft's change of heart.



To understand the lingering enmity that many Linux users have towards Microsoft, you have to go back to the origins of computers in Silicon Valley and the Free Software Movement. The 1970s saw the growth of hobbyist (or 'homebrew') computing kits and hobbyists passing software between each other. But as the movement grew, that free-for-all didn't sit well with fledgling companies that started to supply software.

Ancient grudge

Many were small concerns with staff to pay, who had sunk considerable time into software development and in January 1976, one new company, called Micro-soft, took a stand. The company's 20-year-old general partner, Bill Gates, wrote an open letter to hobbyist that lambasted the software copyright infringement that was rife at the time. "Who can afford to do professional work for nothing?", wrote a very unhappy Gates. "What hobbyist can put three man-years into programming, finding all bugs, documenting his product and distribute for free?" Today, the answer would be a lot of successful free and open source projects, but at the time Gates' grievance was with hobbyists who were copying his company's Altair BASIC, which meant Micro-soft was not receiving any royalties from MITS, the manufacturer of the Altair 8800 computer.

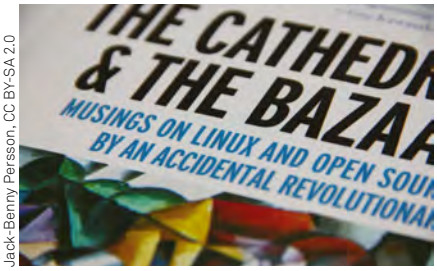
> "'Who can afford to do professional work for nothing?' wrote a very unhappy Gates"

As software development rapidly moved from hobby to big business, the redubbed Microsoft, along with other companies, pioneered a proprietary software business model and by the late 1970s and early 1980s most software was closed.

Four years before Microsoft was founded, Richard Stallman was working in the Artificial Intelligence Lab at MIT and exposed to a community of passionate programmers that shared their software freely. This liberating experience, which ended in 1984, led him to build the legal, philosophical and technological foundations of what we recognise today as the Free Software Movement.

This movement champions the essential freedoms of the user: the freedom to run software, to study and change it, and to redistribute copies with or without changes. This philosophy is backed up by licences, such as the GNU General Public Licence (GPL), that enforce these requirements and cleverly uses copyright to protect free software from being used in proprietary software. Whenever Stallman speaks on the subject of free software he describes software freedom as a question of liberty not price, which is reflected in his often quoted concept of 'free' as in 'free speech', not as in 'free beer'.

It wasn't long before Stallman concluded that his aim of freeing the user from the 'subjugation' of software providers had to start with a UNIX-like operating system that was built from entirely free software. In 1983, he started the GNU Project and without us sliding too far down the rabbit hole, this led to the development of many of the components required for a GNU operating system, such as compiler, debugger, text editor (yes, eMacs), mailers and so on. However, the development of an ambitious microkernel to replace the UNIX kernel, called GNU Hurd, experienced many setbacks – including a three-year wait to see if the preferred Mach kernel would be released with a free software licence. Here



Jack-Benny Persson, CC BY-SA 2.0

QUICK GUIDE

The Cathedral and the Bazaar

A key piece of the puzzle that has ultimately enabled Microsoft to adopt open source development comes from the anthropological observations of software developer Eric S. Raymond in his essay The Cathedral and the Bazaar (CatB), which was first presented in May 1997 at Linux Kongress, Germany, and published as a book two years later.

Raymond's paper attempted to explain how the free software world worked and how it managed to produce high-quality software while flying in the face of perceived best practices at the time.

The essay's title refers to two different styles of software development: the closed 'Cathedral' style that consists of tight objectives, small teams that are generally hierarchically run, and long release intervals; and, in contrast, the open, peer-to-peer 'Bazaar' style that is decentralised, has short release intervals and embraces a constant stream of feedback from contributors who are outside of the formal structure.

The conclusion that was drawn by many reading the paper was that massive, independent peer review was a way to win the software race, and in 1998 it helped spawn the Open Source Initiative.

TIMELINE: HOW MICROSOFT LEARNED TO LOVE LINUX

APRIL 1975

Micro-soft is founded in 1975. A young Bill Gates writes an open letter to hobbyists the following year against the culture of sharing code

FEB 1998

The Open Source Initiative is founded. Eric S. Raymond becomes first president and Bruce Perens drafts Open Source Definition

JUNE 2001

Microsoft CEO Steve Ballmer famously says "Linux is a cancer" in an interview with the Chicago Sun-Times, and is never allowed to forget it

APRIL 2004

The WiX toolset, which is originally developed by Rob Mensching in his spare time, becomes Microsoft's first open source project

Kevin Nixon

Above Without Richard Stallman being marginalised, to a degree, by the popularity of the open source movement, it can be argued that we would not be experiencing such a significant culture change at Microsoft

we skip quickly on to 1991 and a 21-year-old Linus Torvalds releasing his monolithic Linux kernel. The existing software from the GNU Project is rapidly combined with the kernel and that kickstarts the development of the Linux operating system, with the XFree86 project adding a GUI. On the commercial side Red Hat and SUSE release 1.0 versions of their Linux distros. (Note: 'Linux' has become shorthand for what we generally consider the Linux operating systems or Linux distributions we all use, but, technically, it refers to the Linux kernel only. Stallman and influential projects like Debian have always advocated the use of 'GNU/Linux', as the both Linux kernel and GNU software were needed to make a working OS.)

Open source, open door

Without diminishing the importance of other events in Linux's history, such as Apache HTTP server demonstrating a business case for Linux and being monumental in the growth of the World Wide Web, we come to the defining catalyst that has ultimately enabled Microsoft to love Linux: the emergence of open source software development.

In the late 1990s, Netscape was a company that desperately needed a win. It was losing the browser wars to the Microsoft software giant, which was bundling and giving away its Internet Explorer browser for free but not letting anyone near its code. The Cathedral and the Bazaar essay (see p20) supported voices inside Netscape – such as Frank Hecker, a network systems engineer who wrote an internal paper citing CatB – that believed the answer to competing with Microsoft was for the company to release its source code and foster collaboration. To what extent Raymond's essay tipped the balance is unclear, although it remains a highly influential book, but Netscape decided to

Q&A

Interview: Richard Stallman, software freedom activist

Richard Stallman is the father of the free software movement and founder of the GNU Project and the Free Software Foundation.

Do you think that Microsoft has had a change of heart and could become an 'open source' company in the future?
By asking me to discuss Microsoft in terms of 'open source' you're asking me to promote that philosophy. I don't want to say things formulated in terms of 'open source', because I disagree with that philosophy, and I want people to know it. What I care about is free software, respect for the freedom of users. I evaluate Microsoft, or any other company or issue relating to computing, in ethical terms, which is not what 'open source' stands for. Changing from proprietary software to open source makes some difference in practice, but it is not much of a change of heart, and that is why I don't support open source.

Perhaps I should have said "Do you think that Microsoft could become a free software company in the future given that it is opening up in ways we've not seen before?"
"Opening up" doesn't mean support for freedom.

What do you feel are the dangers of what's happening?
It is not a big change. The only specific *new* danger is that people will overstate the significance of it.

2005

Microsoft begins to stick a toe in the open source pond by submitting its Microsoft Community License to the Open Source Initiative for approval

MAY 2005

Microsoft releases F#, a cross-platform, functional-first, object-oriented programming language under an OSI-approved licence, Apache 2.0

JUNE 2006

Microsoft attempts to engage the community with CodePlex, an open source project-hosting portal. In 2017, all projects are moved to GitHub

2006

Microsoft announces it wants to improve support for PHP on Windows Server 2003 and will submit any improvements under the OSI-approved PHP licence

release its Netscape Communicator code in January 1998 under the Netscape Public Licence and founded the Mozilla project. This licence was similar to the Free Software Foundation's GPL, except it allowed Netscape to include proprietary code with what it had freely released.

In the aftermath of a large corporation building a business case for free software, Raymond and Michael Tiemann (co-founder of Cygnus Solutions) and a number of key free software figures got together in Palo Alto, California to strategise how to exploit this influential event. Richard Stallman was not present. It was here that Christine Peterson coined the term 'open source' and it's where we get the Open Source Initiative that promotes a more pragmatic approach to free software development.

> "Open source tends to advocate that non-free and free software should be able to co-exist"

Here was a pivotal crossroad moment. On the one hand we had Stallman declaring an ethical imperative for free software that, in his mind, even outweighed the quality of what was produced with statements such as: "Freedom to cooperate with other people, the freedom to have a community, is important for our quality of life; it's important for having a good society. That, I believe, is more important than having powerful, reliable software". On the other hand, open source went out of its way to promote the economic advantages of volunteers collaborating on open software development and the business case for open-sourcing software.

Open source also avoids the profound hostility towards software patents and intellectual property that is voiced by Richard Stallman and the Free Software Foundation. Essentially, the open source approach doesn't condemn proprietary software and tends to advocate that non-free and free software should be able to co-exist. The Free Software movement's answer has always been to stop using non-free software entirely and permanently.

The emergence of open source may have opened the way for Microsoft, and other big software companies, to consider open source at the end of the 20th century, but MS didn't step through the doorway for a number of years. The new Microsoft CEO, Satya Nadella, may have brought a more gentle yet competitive management style, but, outside of the very fact Microsoft produces proprietary software including the biggest-selling desktop OS in the world, the aggressive strategies employed by both Bill Gates and, in particular, Steve Ballmer in the past haven't fostered trust from the Linux world (see 'FUD and more', right).

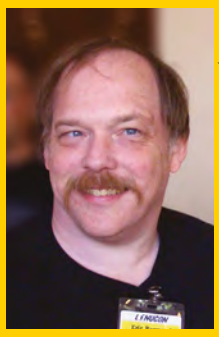
The C word

For instance, whenever anyone writes about Microsoft adopting open source, they always mention the time Ballmer famously described Linux as a "cancer". Steve Ballmer took over as CEO in 2000 and in an interview in June the following year for the Chicago Sun-Times he was asked if Linux and the open source movement was a threat. His response was actually focused on government-funded open source software and how he believed it wasn't available to commercial companies because it meant having to comply with open source licensing: "Linux is not in the public domain. Linux is a cancer that attaches itself in an intellectual property sense to everything it touches. That's the way that the license works."

In fact, it's not difficult to find old interview of Steve Ballmer sticking it to Linux. One we

QUICK GUIDE

FUD and more



Jerone2, CC BY-SA 2.0

The Halloween documents; Embrace, Extend and Extinguish; and FUD. It doesn't take long before a new Linux user encounters these phrases sprinkled in the comments after any article involving Microsoft and open source. These unusual terms stem from a series of confidential Microsoft memoranda that Eric S. Raymond, the author of *The Cathedral and the Bazaar* (pictured above), published on his blog starting in 1998.

The Halloween name refers to the fact that documents were often supplied around the end of October and provided a convenient excuse to use a sinister name. The documents provided insight into the strategies Microsoft was employing to counter open source and Linux, and one of them was a marketing approach called FUD, which stands for Fear, Uncertainty and Doubt. Tactics mentioned involved faking new products and spreading rumours of a competing product's unreliability. More disturbing were the discussions of how to derail a competitor by 'embracing' a public standard; 'extending' the standard and promoting features not supported by a competitor; and once suitably extended, squeeze out and 'extinguish' the competition. As is often pointed out when these terms are trotted out for discussion, the leaks are two decades old now and the world has moved on.

TIMELINE: HOW MICROSOFT LEARNED TO LOVE LINUX

NOV 2006

Microsoft and Novell (owners of SUSE) agree to pay each other for potential IP infringement and Novell agrees to pay royalties on its open source software

OCT 2007

Redmond creates a number of open source licenses and has both the Microsoft Public License and the Microsoft Reciprocal License approved

DEC 2007

After a long legal fight, Microsoft has to give the Samba project proprietary documentation to enable Samba to work smoothly with Windows

JULY 2008

Microsoft starts contributing to Apache Hadoop HBase – an open-source, non-relational, distributed database – when it acquires the company Powerset

Q&A

Interview: Mike Ferris, VP at Red Hat Software

Mike Ferris is vice-president of Technical Business Development and Business Architecture at Red Hat

How has Microsoft's more open approach impacted Red Hat's business?
For years, our mutual customers had been asking that Red Hat and Microsoft work together, so in 2015 Red Hat and Microsoft announced a partnership to help customers embrace hybrid cloud computing by providing greater choice and flexibility deploying Red Hat solutions on Microsoft Azure. Our customers now have more choice, more options and more power. Customers and partners taking advantage of these new capabilities include multinational banks, global retail, large educational institutions, government agencies and systems integrators. They are all looking to Red Hat and Microsoft to provide solutions as they transform their businesses and IT processes.

What have been the most exciting results from collaborating with Microsoft?
The most exciting result from this collaboration is what we are now able to deliver for our customers. Enterprise customers around the world wanted to use Red Hat solutions on Azure, and since our initial partnership was announced in 2015, we've delivered on and expanded this choice. As the Red Hat OpenShift Container platform is becoming the go-to container platform in the enterprise,

we've expanded our work together to help enterprise customers adopt containers. We have aligned engineering and support teams that are delivering innovations across the Red Hat portfolio for Azure, including our unique, co-located enterprise-grade support.

saw recently, filmed at a Gartner Symposium in 2004, sees Ballmer attacking Linux on the security side and suggesting that there were "more vulnerabilities in Linux and it takes longer for the Linux community to get a fix out, so the days of exposure to a vulnerability are longer."

Ballmer goes on to encourage everyone to go to Microsoft's Get-The-Facts marketing campaign site for third-party proof of what he's saying. This highlights another problem with Microsoft: its steady use of FUD, the fear, uncertainty and doubt marketing strategy that attempted to undermine users' confidence in Linux and open source for decades. The trickle release of leaked Microsoft memos from 1998 onwards, dubbed the 'Halloween papers', also confirmed what many already felt: underhanded methods were being employed by the Redmond giant.

> "Attitudes have changed inside Microsoft in thirteen years"

What's Linux good for?

That Gartner Symposium interview in 2004 also reflects how attitudes have changed inside Microsoft in 13 years. When asked whether there were any open source models that work, Ballmer replies: "When it becomes time to ask somebody for a new feature in Linux – who do you talk to?" says Ballmer, in a booming voice that bounces off the back of the hall. Of course, he fails to acknowledge the existence of Linus Torvalds, the Linux Foundation, which opened its doors four years earlier in 2000, or the Open Source Initiative that would have gladly discussed the matter with him. Coincidentally, Ballmer

Below Satya Nadella has a gentler yet competitive leadership style that the tech industry has warmed to

was speaking at the Gartner Symposium on the same day that Ubuntu released 'Warty Warthog' 4.10, the ambitious desktop OS that wanted to be Linux for human beings.

His next comment about intellectual property reflects another reason segments of the Linux and Free Software community still vilify Microsoft: "When it comes time to get indemnification for intellectual property on Linux nobody will give it to you?" asks

JULY 2008

Microsoft joins the Apache Software Foundation as a platinum sponsor and contributes a patch to help PHP work better with SQL Server

2009

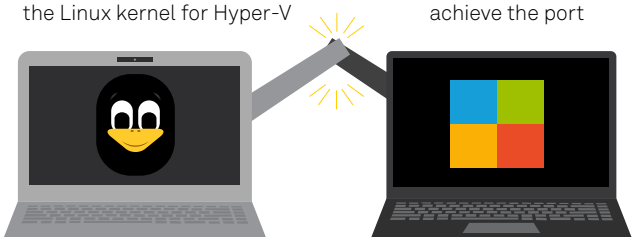
Microsoft says it wants Linux to run as a "first-class citizen" on its virtual servers and contributes over 20,000 lines to the Linux kernel for Hyper-V

NOV 2011

The first stable release of Node.js for Windows, as Microsoft works with Joyent and Node.js author Ryan Dahl to achieve the port

2012

The company releases ASP.NET MCV, Razor and Web API under Apache 2.0 "to enable a more open development model", says Microsoft's Scott Guthrie



a concerned Ballmer. Today, it's estimated that Microsoft makes at least $3 billion in annual royalty payments from its software patents portfolio and its hounding of Linux companies for patent agreements has been deeply unpopular. However, in what has mostly been viewed as a positive step, it dropped some long-running lawsuits in recent years.

It may be surprising to note, given what Ballmer was saying at the time, that 2004 was also the year that Microsoft first released open source software. "Some of the open source stuff, that actually began under Ballmer, believe it or not," says Martin Woodward, Principal Program Manager at Microsoft who introduced Git to the company (interview, p27). "And in some ways he doesn't get enough credit. I mean there's people that aren't going to forgive him for the 'Linux is a cancer' comment... But a lot of the One Microsoft stuff [which came later when the company focused on devices and services] began under his watch."

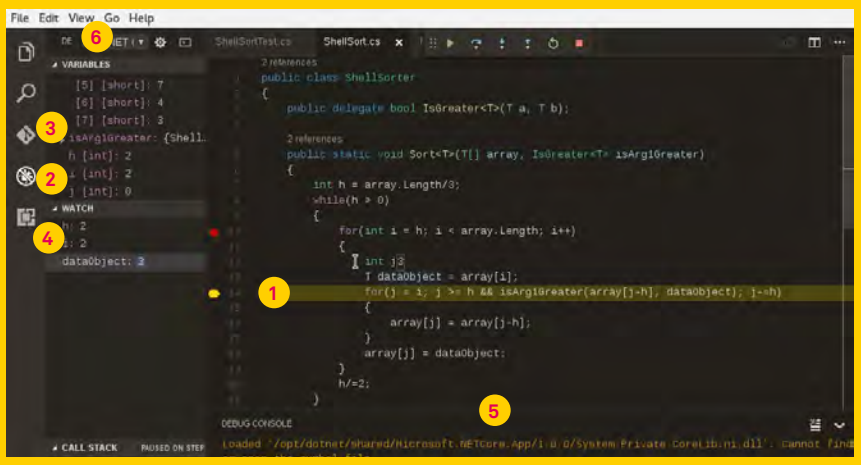
> "It's estimated that Microsoft makes at least $3 billion in annual royalties"

WiX came first

The first open source software was a toolset called WiX and that was used to build Windows Installer packages from XML. It was released under a Common Public licence (CPL) which is approved by both the Open Source Initiative and the Software Freedom Foundation. The CPL allows proprietary software to link to a library under CPL without being forced to adopt the same CPL licence. The person to thank for that is Rob Mensching, the original author, who worked on the project in his spare time. On his personal blog he says that back in 1999 and 2000, he didn't feel that many people inside Microsoft understood what the open source community was really about and "wanted to improve that understanding by providing an example".

Since then – as you can see from the timeline running throughout the feature – it's been a steady trickle of Linux-related announcements and open sourced projects, with notable surges such as 2012 when Microsoft moved into the top 20 committers to the Linux kernel, while it was working on the kernel drivers for its Hyper-V virtualization hypervisor. That trickle turned into an open source river from 2014 onwards, straight after Satya Nadella took the helm of the Redmond giant.

"I think it helps if you understand why we're changing," says Martin Woodward, who introduced Git to the company. "We're still a business, it's just we deliberately changed our business to try and make it a lot more open-source friendly. Azure really helped, so now we make money by selling access to Linux – we sell an awful lot of



REASONS TO TRY VISUAL STUDIO CODE

1. Intellisense – This is a feature that supplies smart completions based on variable types and function definitions.
2. Debugging live – You can attach to running apps and debug with break points, call stacks and an interactive console.
3. Git & other SCM support – Review diffs, stage files and make your commits direct from the editor.
4. Extensions – Install extensions to add new languages, themes, debuggers and to connect to additional services.
5. Pipe out – 1.19 adds support to pipe the output of a terminal command directly into VS Code and have it open in an editor.
6. Logging – Also in 1.19, VS Code now creates activity log files which can help diagnose unexpected issues.

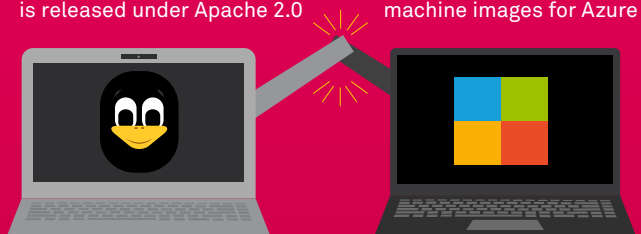
TIMELINE: HOW MICROSOFT LEARNED TO LOVE LINUX

2012
A subsidiary of Microsoft called Microsoft Open Technologies, Inc., is announced to "advance the company's investment in openness"

OCT 2012
A three-year project in the making, TypeScript – an open source programming language and JavaScript superset – is released under Apache 2.0

JAN 2013
Microsoft Open Technologies, Inc. opens VM Depot, a community-driven repository of Linux and FreeBSD virtual machine images for Azure

FEB 2014
Satya Nadella is appointed CEO of Microsoft. Announces Cloud First, Mobile First strategy and dethrones Microsoft Windows as beating heart of the business



Linux Foundation, CC BY 2.0

COMMENT

Jim Zemlin, Executive Director of The Linux Foundation

"Like virtually every software company in the world, Microsoft realises open source is the way most software is created today, and will continue to be for the foreseeable future. Microsoft has made huge strides and actively supports, relies on and contributes to numerous open source projects including Cloud Foundry, the Cloud Native Computing Foundation, the Core Infrastructure Initiative, Linux, Node.js, the Open API Initiative, the TODO Group and more. This means more quality open source code, which benefits the entire community."

Linux virtual machines," he adds.

Microsoft says that one in three VMs hosted on Azure are Linux. "But probably more importantly, we want customers to host their workloads on our cloud."

This means that any Linux-based software has to work really well in Azure, which also means as smoothly as possible under Hyper-V, and this is exactly why Microsoft produces so many Linux kernel fixes. "Because we sell a lot of Linux it has to work well in our data centre. If we can improve its performance by .05% [...] that's a massive cost saving for us which justifies us having engineers working full- time improving the [Linux] kernel," he says.

However, Woodward adds that the biggest change in the company was when they started explaining things to the managers in business terms. He says that because Microsoft does open source, it's a lot easier to recruit people now. It's a simple truth, particularly in the tech industry, that well-regarded firms find it easier to recruit top-notch talent. "It improves the perception, the PR, of Microsoft. But those are sort of kumbaya, ancillary benefits, those aren't the business reasons. Those reasons are that we want to host stuff more effectively and we want to ship more stuff and so we build lots of these open-source projects."

Turning the ship

Microsoft's reasons for contributing code back are solidly grounded in open source's business-orientated philosophy: "Firstly, some projects, like the Linux kernel, are GPL-licensed, so we have to," says Woodward. "That's in the licence, and if we want people to respect our licenses for commercial software then we have to respect open source licenses." But the main reason why Microsoft contributes, even for MIT- and Apache-licensed code, is the issue of having an internal fork of an open source project. "The further you diverge from that fork," says Woodward, "the harder it is – if you're contributing locally and not pushing back upstream – to get the changes from upstream back into your codebase. Over time, then, you end up with this completely divergent real fork of an upstream open source project and you are no longer able to get all the contributions from the rest of the community into your version, so you may as well have just built it yourself."

The other reason, as with the Linux kernel, is to contribute to make the open source software run faster inside of Azure, or Windows Server or wherever. The question then becomes, according to Woodward, "Why wouldn't we want as many people as

Q&A

Dustin Kirkland, VP, Product Development at Canonical

What collaborative projects with Microsoft have excited you the most?
We've worked closely with the Azure team to build a very specifically optimised Linux kernel that improves performance and security in the Azure cloud. We're also excited about Ubuntu as the basis for Microsoft's Kubernetes Service. On the desktop side of things, it's impossible to underestimate how many millions of Windows Desktop users are excited about having a native Ubuntu/Bash experience built right into their desktop, with access to thousands of binary Ubuntu packages one simple `apt-get` install away.

With IoT and connected devices, we can't say much here yet, but 2018 will be an exciting year for Ubuntu Core, Snap packaging, and how those interact with the Microsoft world...

Given Microsoft's history of suing or settling with companies over Linux software patents, is Canonical likely to make an agreement with Microsoft to avoid future litigation?
Microsoft's engineering investment in the Windows Subsystem for Linux (WSL, see p26), and enthusiasm to bring Ubuntu into the Windows Desktop WSL from March 2016 to present, signal to us and the world a different posture from Microsoft around Linux. And that's just the tip of the iceberg. Linux in Azure, SQL Server on Linux, Visual Studio on Linux – these aren't the actions of the Microsoft which Canonical first encountered when it created Ubuntu in 2004.

OCT 2014
Microsoft's new CEO Satya Nadella declares that "Microsoft Loves Linux" and doesn't throw any chairs. Porcine aviation confirmed

NOV 2014
Redmond giant open-sources the modular development stack of .NET, including ASP.NET, .NET compiler, .NET core runtime, framework and libraries

SEPT 2015
The company releases a cross-platform modular operating system for data center networking built on Linux called the Azure Cloud Switch

NOV 2015
Microsoft releases Visual Studio Code, a rich, cross-platform source code editor, under the MIT Licence with support for extensions

possible to have those changes, because we still benefit?"

The sheer number of open source projects that Microsoft has released is vast, but we'll highlight a number. Unveiled at the Build 2016 keynote, Windows Subsystem for Linux (see quick guide, below) came out of beta with the Windows 10 Fall Creators Update. You may have heard that you can install Linux distributions via the Windows Store with it, but as they aren't full-blown Linux distros you may be wondering what the point is. "The main reason for doing it was for devs, "says Martin Woodward, a Principal Program Manager at Microsoft, "Basically, we want Windows to be an awesome desktop for a developer, so you can run all the apps that you have to run, Office and Outlook and things. But then if you want to apt-get something or use Vim or Emacs it's there and it's proper Bash that you're typing out."

Code, lots of OS code

Microsoft has also released an open source, modularised form of the .NET technology base called .NET Core that is cross-platform and offers support for a variety of Linux distros. The companion, .NET Core SDK, provides the command-line tooling to create simple .NET scaffold code, download project dependencies, compile, package and publish.

Above TypeScript and its ability to enable application-scale JavaScript has been very popular

However, to really benefit from .NET Core development on Linux, you need an editor that can support the iterative code-edit-debug cycle. Enter Visual Studio Code, which was open-sourced in November 2015. This is a rich, cross-platform editor is highly extensible, supports several programming languages and includes a debugger, and has built-in Git client support. True to its open source roots, Visual Studio Code provides embedded Git control for standard client actions. It also has built-in debugging support for the Node.js runtime.

The general philosophy of the editor is to make features discoverable from a keyboard-centric, auto completion-enabled command palette, rather than deep menu hierarchies rendered in a complex UI. These features are either built-in or can be dynamically loaded into the code editor via extension or injected via JSON-based files. All these projects certainly making friends in the Linux

QUICK GUIDE

Windows Subsystem for Linux

Interoperability is the key focus of WSL, and Rich Turner, Microsoft's senior program manager of WSL and Windows Console, describes it as a way to "run modern-day development tools on Windows." The project comprises Bash on Ubuntu on Windows, which is the user-mode only portion of Ubuntu Linux, created especially by Canonical. Underneath is WSL itself and this subsystem sits within the Windows kernel and emulates the behaviour of a Windows kernel implementation. This allows Microsoft to run the user-mode segment of Ubuntu without having to modify the Linux tools you want to use (such as apt, ssh and many others) and enables Linux binaries to run unmodified directly on Windows. For now Bash and WSL are focused on command-line tools, but if you're a developer that deals with Windows it's still an exciting project.

Above Windows Console is also being developed to have better compatibility with popular Linux tools

TIMELINE: HOW MICROSOFT LEARNED TO LOVE LINUX

DEC 2015

Microsoft brings Debian GNU/Linux to Azure cloud through a partnership with credativ and offers both Debian "Wheezy" 7 and "Jessy" 8

NOV 2015

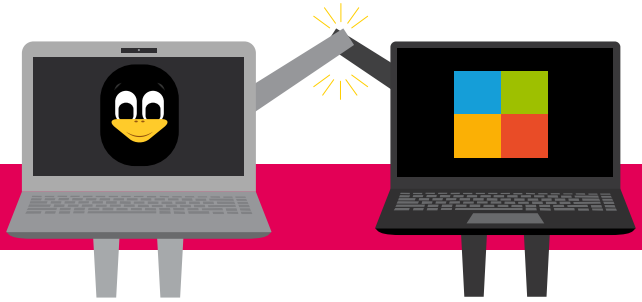
Red Hat and Microsoft make a deal to bring Red Hat Enterprise Linux (RHEL) to its Azure cloud. Both also agree not to sue each other over patents

JAN 2016

Ubuntu Linux is previewed for the first time on Azure. Microsoft now has all leading Linux distributions available for its Azure cloud computing platform

AUG 2016

Windows 10 introduces Windows Subsystem for Linux that makes it possible to run Bash and other Linux tools in an Ubuntu-based user mode environment



community. Joshua Strobl, Communication Manager for the Solus Project, says he's had a positive experience working with Microsoft on the Visual Studio Code project (as the package maintainer and integrator for Solus): "Whether it's providing a first-class developer experience with Visual Studio Code, enabling application-scale JavaScript with TypeScript or developing an agnostic .NET runtime, Microsoft has shown clear commitment to open source and engaging the open source community in a constructive and meaningful manner."

Another hit that wasn't quite expected by Microsoft was how much TypeScript has taken off since its release. This was another "classic" from Anders Hejlsberg, the father of C# and TurboPascal, according to Woodward. "At the time I was thinking 'Who's going to adopt this Javascript variant from Microsoft?'

> "The question then becomes 'Why not open source it?', rather than 'Why should we?'"

and then all of a sudden the Angular.js team pick up on it and it starts getting a bit of adoption as a type-safe Javascript for large applications. Shows you what I know." Again, this was something that was built for Microsoft's internal use, then open sourced because, as Woodward says, "Why not? Once it's open source we get a lot of people contributing, but also a lot of tooling that knows about it and Angular hooked into it. The question then becomes 'Why not open source it?', rather than 'Why should we?'".

Whether you can find it in your heart to support Microsoft, there's no question that the level of open source contributions in only three years under Nadella's leadership are an impressive achievement.

Q&A

Interview: Martin Woodward, Microsoft

Woodward is the Principal Program Manager at Microsoft, working on Visual Studio Team Services, and is the person behind the migration of 65,000 Microsoft engineers to Git. Woodward joined Microsoft in 2009 after working at a small startup that created plug-ins for Eclipse. The business built a plug-in for Microsoft's Team Foundation Server and Microsoft subsequently bought the company.

When you first joined were you shocked at how development took place? Obviously coming from a small team to a massive organisation, there's going to be some amount of culture shock?
It was a system that had been built up over two decades, and it was very much individual groups working on their own thing. There wasn't any one way of building anything, that was one side of it. But I was working in a startup where we were just shipping code all the time. But at Microsoft there'd be these nine month periods where people would build stuff, and in those nine months there'd only be six weeks where people where actually coding. My initial thought was "What's happening the rest of the time?". Trying to figure those things out was a bit of a shock. The first course I went on – besides the standard courses, you know, standards of business conduct – was one that allowed me to approve the hologram that went onto the DVD which was then packaged in a shrink-wrapped box and shipped. "We ship discs?", I remember being shocked.

I think the three main culture changes that I saw were everyone going from having their own engineering systems to 'One Microsoft'; using more open source and contributing back to open source and shipping always-on services. Previously, it had been shrink-wrapped software, which gives you this mindset of optimising for not failing. With services you're always optimising for how quickly can you fix something when it fails. That's a very different mentality altogether.

So as well as bringing some open source ideas to MS, you also moved 65,000 engineers onto a single Git platform. What made you choose Git?
This was just after I joined, so early 2010. Distributed version control was a disrupter in the version control market; it works completely differently to centralised systems. Being Microsoft, 'Buy, build or use open source' were the three decision points. We could've bought a commercial company doing that, which we looked at. We could've built our own, which a lot of people at MS would've preferred to do because 'Not Invented Here' was still very much part of the mindset. Then there were the open source options, which at that time were Mercurial and Git. Everyone would've told you Mercurial is a better and more useable version control system, and works better on Windows.

NOV 2016
Microsoft joins the Linux Foundation as a Platinum sponsor and welcomes Google to the .NET Foundation and the Steering Group

JULY 2017
The first release candidate of SQL Server 2017 for Linux is announced and will allow a consistent data platform across Windows and Linux

DEC 2017
The company open-sources the Virtual Kubelet that connects Kubernetes clusters to Azure Container Instances and unveils Kashti, a visual dashboard

DEC 2017
Microsoft adds OpenSSH client into the Fall Creators Update for Windows 10 and says that it will contribute to the OpenSSH community

That was the conventional logic at the time. But when we started digging in, Git was being used as the network protocol for distributed version control, between DVCS providers, but when you were deploying to the cloud we were seeing Git protocols being used more and more. When you dig into the Git network protocol, it's actually a very, very thin shim over the .git directory in your filesystem. So we thought "Huh. That's interesting". So the reasons for using Mercurial over Git boiled down to, it works better on Windows. But if we got the Windows team to use Git then we could fix that. At the time it looked maybe like Git might win, back in 2010 it was hard to tell, but luckily we bet on the right one. Today something like 70 per cent of developers are using Git, so we made the right choice.

Above Martin Woodward says that one of the things he has to do is take the swear wordsout of Linus Torvalds' code, as "he's a very sweary kind of guy"

How did you manage to get Git to handle such a huge codebase?
Well, we did some changes on the server to handle that load. But we also introduced this thing called Git Virtual Filesystem (GVFS), which is an extension onto Git that's open source, obviously. It's a filesystem with a directed acyclic graph. It's kind of weird because GVFS makes Git not distributed anymore. It virtualises a lot of stuff in the .git/ directory. The idea is that when you clone a Git repo, instead of pulling everything locally it streams it on demand as you request parts of it. That's how we can scale up to these massive repos. So when you clone one all you're really doing is copying a bunch of pointers. As you try and access stuff it's dynamically pulled in and a bunch of clever behind-the-scenes stuff happens. But it can also pull in from a local cache. So you can use one server as your golden repo, that everybody's syncing with, but then when you do a clone it's smart enough to pull down the binary files, say from your local cache in Japan. This led to some oddities, such as when the team in Japan could clone the Windows source code quicker than the team in Redmond, because the team in Redmond had to go all the way across campus to get their cache, but the team in Japan just had to go to a server that's in their cupboard.

> "Hovering over the button to publish, we were all excited: 'Will it work, is it going to work?'"

You open sourced a C# compiler too?
Yep, Anders Hejlsberg was the guy that open sourced that, and this is a good illustration of the MS open source journey. I was in charge of Codeplex at the time, which is our open source hosting platform that we've had forever. He was going to open source Roslyn, and that's not something you can practise very often. So hovering over the button to publish, we were all excited – "Will it work, is it going to work?". And we did it, and it did, and that was great, but what we were actually doing behind the scenes was publishing to an open source hosting platform. So nowadays we wouldn't classify that as true open source. Even though it's under an open source licence, and the source is available, the engineers were all working in the background on TSS (the internal version of Visual Studio Team Services), they were all working in a different place. The community got to see a version, and they can do a pull request against that version, then the engineers have to pull it over, internal merge it, do some magic and then push it back out again. The big shift in terms of contribution was when we moved Roslyn, and this corresponded to when we moved it over to GitHub as well, but rather than having the team working on it in one place and the community working on it in another, we moved the whole team over to GitHub. So they're working on the same repo as the whole community, so if they want to do a change to Roslyn they have to go through the same pull request process that you would have to go through.

So as engineers there's no secrets, you're all contributing to the same repo. That changes a lot of culture in terms of how we document things, making sure we get a lot better about doing asynchronous communication rather than watercooler-type communication. That then helps the culture inside the business. I mentioned going from separate silos into one engineering team. Now we're all using the same engineering system and we're all using Git as well, if I want to make a change to, say, Notepad – to finally add support for Unix line endings, this is actually something I've seriously looked at a few times – I can send a pull request to the Windows team with that change. I can see all their code, so it's an inner source model. So through our interactions with open source, the engineering teams are learning this pull request culture and workflow. The way we handle dependencies has changed a lot because of how we work with open source.

TIME TO STEP OFF THAT TREADMILL

With so many demands from work, home and family, there never seem to be enough hours in the day for you. Why not press pause once in a while, curl up with your favourite magazine and put a little oasis of 'you' in your day.

PRESS PAUSE
ENJOY A MAGAZINE MOMENT

To find out more about Press Pause, visit:
pauseyourday.co.uk

FREE DVD UBUNTU 17.10 SPECI
Linux User & Develo
42 PAGES OF GUIDES
MASTER UBUNTU 17.10
All the new features! PLUS the best extensions, customisation tools and alternative spins
FREE DVD! LATEST LINUX MINT!
www.linuxuser.co.uk
Linux User & Developer
THE ESSENTIAL MAGAZINE FOR THE GNU GENERATION
SPECIAL REPORT
MICROSOFT
How I learned to stop worrying and love Linux
Is Microsoft a friend or foe?
It's time for you to decide
INSTALL 2 RASPBERRY PI DISTROS TODAY!
Pi-optimised Raspbian Stretch Lite and DietPi v150
LINUX MINT
CINNAMON 18.3
The best Windows-like desktop OS for the Linux user
PLUS ZORIN OS 12.2 CORE
Now with Wine 2.0 for better Windows application support
43 PAGES OF GUIDES
> Security: sniff out reliable exploits
> Master GNU Make
> Java: Add AI
HUGE ROUNDUP
Get the perfect single board computer
21 tiny boards on show
ALSO INSIDE
TRY 2 UBUNTU SPINS
Data recovery • File system & cloning • Security analysis
IN-DEPTH GUIDE
The future of programming
42 PAGES OF GUIDES
> MQTT: Master the IoT protocol
> Security: Intercept HTTPS
> Essential Linux: The joy of Sed
Get into Arch Linux
ALSO INSIDE

~~£6.49~~
£5.19
per issue

Subscribe and save 20%*

Every issue, delivered straight to your door

Never miss an issue
13 issues a year, and you'll be sure to get every single one

Delivered to your home
Free delivery of every issue, direct to your doorstep

Get the biggest savings
Get your favourite magazine for less by ordering direct

What our readers are saying about us…

"I've only just found out about this magazine today. It's absolutely brilliant and exactly what I was looking for. I'm amazed!"
Donald Sleightholme via Facebook

"@LinuxUserMag just arrived by post. Wow what a fantastic issue! I was just about to start playing with mini-PCs and a soldering iron. TY"
@businessBoris via Twitter

"Thanks for a great magazine. I've been a regular subscriber now for a number of years."
Matt Caswell via email

Pick the subscription that's right for you

MOST FLEXIBLE

Subscribe and save 20%

✔ **Automatic renewal – never miss an issue**

✔ **Pay by Direct Debit**

Recurring payment of £33.75 every six months, saving 20% on the retail price

DIRECT Debit Instruction to your Bank or Building Society to pay by Direct Debit

Originator's reference 7 6 8 1 9 5

Name of bank

Address of bank

Postcode

Account Name

Sort Code **Account no**

Please pay Future Publishing Ltd Direct Debits from the account detailed in this instruction subject to the safeguards assured by the Direct Debit guarantee. I understand that this instruction may remain with Future Publishing Ltd and, if so, details will be passed on electronically to my Bank/Building Society. Banks & Building Societies may not accept Direct Debit instructions for some types of account

Signature **Date**

GREAT VALUE

One year subscription

✔ **Great offers, available worldwide**

✔ **One payment, by card or cheque**

A simple one-off payment ensures you never miss an issue for one full year. That's 13 issues, direct to your doorstep

☐ UK £67.50 (saving 20% on the retail price)

☐ Europe €88.54 ☐ USA $112.23 ☐ Rest of the world $112.23

Pay by card or cheque

Pay by Credit or Debit card

☐ VISA Visa ☐ MasterCard Mastercard ☐ AMERICAN EXPRESS Amex

Card number

Expiry date /

Pay by Cheque
I enclose a cheque for £

Made payable to Future Publishing Ltd

Signature **Date**

Your information

Name

Telephone number **Mobile number**

Email address

☐ Please tick if you want to receive any communications from Future and its group companies containing news, special offers and product information.

Address

Postcode

Please post this form to
Linux User & Developer Subscriptions, **Future Publishing Ltd, 3 Queensbridge, The Lakes, Northampton, NN4 7BF, United Kingdom**

Order securely online **www.myfavouritemagazines.co.uk/sublud**

Speak to one of our friendly customer service team
Call **0344 848 2852**

These offers will expire on
28 February 2018

Please quote code **LUDPS17** when calling

*Prices and savings are compared to buying full-priced print issues. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Payment is non-refundable after the 14-day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: bit.ly/magtandc Offer ends 28 February 2018.

John Gowers
is a university tutor in Programming and Computer Science. He likes to install Linux on every device he can get his hands on, and uses terminal commands and shell scripts on a daily basis.

Resources

A terminal running the Bash shell

GNU Make (included in most Linux distributions) www.gnu.org/software/make

PART ONE

An introduction to building programs with GNU Make

The GNU Make program can greatly simplify and automate the process of building a large software project

Welcome to this new Linux Essentials series, in which we will be learning how to use the program GNU Make. If you've ever installed software from source, then you have probably run the `make` command to install the program. Make dates from 1976, and has become almost universal as a tool for building large software projects, particularly those written in C and C++, such as GCC and the Linux kernel. Although some languages now have their own build tools, Make is still an important tool for any Linux programmer to know.

Make is a versatile tool that can be used with any language that supports a command-line compiler. We're going to learn how to write and run 'makefiles' for many different sorts of software project. The basic concepts of Make are simple, but we'll also learn a number of different tricks that we can use to keep our makefiles more compact and maintainable – including features specific to GNU Make, the version of Make present on Linux systems.

Compiling a large piece of software can be a complicated task. Often, there are multiple ways to compile source files even within the same language. Moreover, the build order of a project is often important: certain source files need to be compiled before others.

Since we don't want to keep typing the same commands every time we want to build our project, it makes sense to automate the process. The simplest way of doing this is by writing a shell script. For example, if we had written a simple DVD player program that depended on C source files `player.c`, `screen.c` and `dvd_read.c`, then we could write the following shell script:

```
#!/bin/sh
gcc -c player.c screen.c dvd_read.c
gcc -o player player.o screen.o dvd_read.o
```

As we saw in a recent series, shell scripts are very powerful, so it should be possible to automate quite complicated project builds using a shell script.

There is a much better way, though, which is to use GNU Make. Make is a special language specifically for writing project builds from source, and GNU Make is

the version included in most Linux distributions. Builds scripted using GNU Make are much more maintainable, configurable and compact than builds scripted using shell scripting.

Understanding dependencies

Central to the operation of GNU Make is the concept of a dependency. When we compile a program, we typically need one or more source files, and possibly other resources as well. The specific files that we need to carry out a particular build are known as the dependencies of that build.

Often, dependencies will be attached to a particular file. These are the files that are required in order to build that file. For example, in the code above, the dependencies of the **player** program are the object files **player.o**, **screen.o** and **dvd_read.o**. Meanwhile, the file **player.o** has the source file **player.c** as a dependency. It may in addition have other dependencies that are not mentioned in the command that we use to compile the file: for example, C programs often use header files, marked with **.h**, which are included directly inside the source files.

> Make will only recompile a file if one of its dependencies has been modified since the last build

Dependencies are often source files of some kind, but all sorts of files can be dependencies. For example, in large projects it is common to include source files in a directory called **src** and the compiled binaries in a separate directory called **bin**. In that case, the directory **bin** is a dependency of the binaries, since it needs to be created in order for them to have somewhere to live.

Creating your first makefile

In order to script a GNU Make build, we need a special file called a makefile, which is a bit like a script for running the build. Makefiles can have one of three names: **Makefile**, **Makefile** or **GNUMakefile**. We recommend that you call your Makefiles **Makefile**, with a capital M, in order to make them appear higher up in search results.

We run a makefile using the command

```
$ make
```

If we have a makefile that doesn't have one of the three special names, we can run it using **make -f my_oddly_titled_Makefile**. Inside the makefile we put our script. A simple makefile that compiles our C program is shown in **Figure 1**, above.

From looking at **Figure 1**, we can deduce certain features of how a makefile is set up. The file consists of a series of rules, which are usually files that we need to build in some way. The first rule is the main rule, the program that we eventually want to create. Each rule is

Figure 1

```
player : player.o screen.o dvd_read.o
  gcc -o player player.o screen.o dvd_read.o

player.o : player.c player.h screen.h dvd_read.h
  gcc -c player.c
screen.o : screen.c player.h
  gcc -c screen.c
dvd_read.o : dvd_read.c player.h
  gcc -c dvd_read.co
```

Above It's easy – and worthwhile – to write Makefiles for simple projects as well as more complicated ones

followed by a colon and then a list of the dependencies required to complete that file. For example, in order to build the program **player**, we need the object files **player.o**, **screen.o** and **dvd_read.o**. In order to build the object file **player.o**, we need the source file **player.c** and the header files **player.h**, **screen.h** and **dvd_read.h**.

Immediately below each rule line is a tab-indented line called a recipe, which is the shell command that we use to build the rule. When Make tries to build the program **player**, it will run that command.

It may seem superfluous to include complete lists of dependencies for each rule, since all we are really interested in is the shell command that we will run to build the files. In fact, the dependencies serve a very important purpose. To illustrate this, suppose we're trying to compile the **player** program. If we try to run the command

```
$ gcc -o player player.o screen.o dvd_read.o
```

without creating the **.o** files first, we run into an error. Including the list of dependencies means that Make, before trying to build the **player** program, will make sure that the **.o** dependencies have been created first. If they haven't, or if they have been modified, then Make will look for an appropriate rule in order to create them. In our case, we have marked **player.o** as a dependency of **player**, and so Make will run the **player.o** rule before it runs the **player** rule, if that **.o** file does not exist or has been modified.

Make and the dependency tree

What we mean by 'modified' is a very important point, and one that showcases a big advantage that Make has over shell-scripted builds: Make will only recompile a file if it, or one of its dependencies, has been modified since the last build. This can greatly save time when recompiling a project. In order to understand how this works, we need to take a closer look at the workflow that Make follows when it parses a makefile.

Make works recursively: when it reaches a rule, it first tries to resolve all its dependencies before carrying out the rule itself. Make runs over all the dependencies, deciding whether each one is up to date.

Mind the gap now

A common mistake when writing makefiles is to omit the tab before a recipe definition, or to write two spaces instead. Unlike other languages, Make is very particular about white space. A recipe must always be preceded with a tab character, or Make will complain and exit with an error, or will not process the Makefile as you expect. Because a recipe can span multiple lines, Make uses the tab to determine which lines are part of that recipe.

Silencing in recipes

By default, Make displays each command in the recipe before executing it, which can be useful for debugging. In order to suppress this, we can silence a line of the recipe by preceding it with the at sign, @. We might, for example, want to show some text while doing our build, for which we could use the command `@echo my_text`. If we omit the @, the echo command itself is displayed as well.

There are two types of dependencies. The first is a dependency that has no attached rule, typically some kind of source file. If Make encounters such a dependency, then it will check to see whether the timestamp on that file is more recent than the timestamp on the file that we are trying to build. If it is, this indicates that the source file has been modified more recently than the target file, and in such a situation Make will decide that it needs to run the recipe for the target file in order to incorporate the changes.

The second type of dependency is a rule that occurs somewhere else in the Makefile. For example, the `.o` dependencies for the `player` rule all have their own rules further down. In that case, Make recursively processes each of these rules before processing the target rule.

If none of the rule dependencies require recompilation and if none of the source files have been modified, Make will not recompile the target. Otherwise, it will run the recipe for the target rule in order to create the file.

To illustrate this, let's see what might happen if we decided to modify the functionality of the way our DVD player displayed on screen. We might make a few changes to some of the functions in the `screen.c` file.

> "As in any language, code repetition is something we want to avoid in Makefiles"

When we run the command `make`, Make starts off with the first rule in the file: `player`. Before it does anything with that rule, however, it first needs to process each of the dependencies.

The first dependency is `player.o`, which is a rule dependency. The dependencies of `player.o` are all source files; since none of them has been modified more recently than `player.o`, Make skips this rule without doing anything.

The second dependency is `screen.o`, which is another rule dependency. `screen.o` has two dependencies of its own: `screen.c` and `player.h`, which are both source dependencies. Since we have just been making some changes to `screen.c`, its timestamp is more recent than that of `screen.o`, and so Make will run the recipe `gcc -c screen.c` in order to rebuild the object file.

The last dependency is `dvd_read.o`. Like `player.o`, all the dependencies of this rule are source files and none of them have been modified recently, so Make will skip this rule.

Now that Make has processed all the dependencies for `player`, it can decide whether or not to run the recipe for that rule. Since we ran the rule for `screen.o`, it means that that file has now been modified more recently than `player`, and so Make will therefore run the recipe `gcc -o player player.o screen.o dvd_read.o` in order to build the program.

If we had used a script we would either have had to settle for compiling all three C files, or we would have had to write a complicated script that tested the timestamps of each file in order to work out whether it needed to compile them or not. With Make, this is all done for us!

Our example is quite simple, but larger projects can develop a more complicated dependency tree, in which a rule has several dependencies, each of which has dependencies of its own, each of which has further dependencies and so on. The 'leaves' of this tree are the source files, which have no dependencies of their own. If any of these source files is absent, the makefile cannot run.

One thing to watch out for is circular dependencies. If we try to run the following makefile:

```
bee : flower
flower : bee
```

then we might expect Make to run forever. In fact, it is clever enough to detect what is happening and will skip these dependencies:

```
$ make
make: Circular flower <- bee dependency dropped.
make: Nothing to be done for 'bee'.
```

Figure 2 (left) demonstrates some pseudocode that might help you better understand how Make works if you're still not quite sure.

Make variables

In the Makefile in **Figure 1**, we had to write the list of object files (`player.o screen.o dvd_read.o`) twice, once on each line. As in any language, code repetition is something we want to avoid in Makefiles: if we wanted to add an extra `.o` file to the dependencies of `player`, then we would need to modify both lines.

Luckily, Make provides the concept of variables, which can store text and can be used as many times as we like in the file. In our case, we could add the following to the top of our Makefile:

```
OBJECT_FILES = player.o screen.o dvd_read.o
```

Figure 2

```
function make(rule)
  var needToCompile = false
  for (d : rule.dependencies)
    if (isRule(d))
      needToCompile |= make(d)
    else if (exists(d))
      needToCompile |= (d.date > rule.date)
    else
      error "Dependency " + d + "does not exist!"
      exit
  if (needToCompile)
    exec(rule.recipe)
  return needToCompile
```

Below Some pseudocode for a simplified version of Make. This code does not behave exactly as Make does, but it's a good first approximation

We would then be able to change the first rule to:

```
player : $(OBJECT_FILES)
    gcc -o player $(OBJECT_FILES)
```

Notice that we reference variables by using a dollar sign $ and round brackets. If you are used to shell scripting, this might look a bit strange, since shell-scripting normally uses ${...} for variables and $(...) to get the output of a program. The fact that we can put spaces around the equals sign = when writing to a variable is another difference from shell scripting. It is important to remember that Make is a separate langauge from shell scripting, and that although it sometimes looks similar, it may have different syntax sometimes.

Recipes, on the other hand, are written in shell syntax. We'll come back to this point soon.

Globs and substitution

Makefiles support exactly the same globbing functionality as the shell. For example, if we include

```
*.o
```

in the definition of a rule, it will expand to give the list of all .o files in the directory containing the Makefile.

One thing we might try would be to replace the list of .o files with *.o:

```
player : *.o
    gcc -o player *.o
```

This will work as long as the directory containing the makefile contains exactly the object files that we need. But what if we delete the file dvd_read.o? Then *.o will expand to player.o screen.o, and the file dvd_read.o will not be rebuilt.

One possible solution to this problem is through substitution. We should not rely on the .o files always being present; indeed, we should be able to delete and rebuild them whenever we want. However, we can rely on the source .c files being present: if they are deleted, then we have no hope of compiling the program again. So one solution is to say that the object files we want should be the same as the source C files, but with the extension .c changed to .o.

For this, we will use the Make function patsubst. If we write the following:

```
OBJECT_FILES = $(patsubst %.c,%.o,$(wildcard *.c))
```

then Make will populate the OBJECT_FILES variable with the names of the C source files after the .c has been changed to .o.

There are a few parts to this line. First are the functions patsubst and wildcard. In order to call a function in Make, we use the same $(...) syntax that we use for variables. The wildcard function performs glob expansion on its argument: glob expansion is automatic

```
uninstall-binPROGRAMS:
  @$(NORMAL_UNINSTALL)
  @list='$(bin_PROGRAMS)'; test -n "$(bindir)" || list=; \
  files=`for p in $$list; do echo "$$p"; done | \
    sed -e 'h;s,^.*/,,;s/$(EXEEXT)$$//;$(transform)' \
        -e 's/$$/$(EXEEXT)/' \
  `; \
  test -n "$$list" || exit 0; \
  echo " ( cd '$(DESTDIR)$(bindir)' && rm -f $$files )"; \
  cd "$(DESTDIR)$(bindir)" && rm -f $$files

clean-binPROGRAMS:
  -test -z "$(bin_PROGRAMS)" || rm -f $(bin_PROGRAMS)

clean-checkPROGRAMS:
  -test -z "$(check_PROGRAMS)" || rm -f $(check_PROGRAMS)

loadavg$(EXEEXT): $(loadavg_OBJECTS) $(loadavg_DEPENDENCIES) $(EXTRA_lo
  @rm -f loadavg$(EXEEXT)
  $(AM_V_CCLD)$(LINK) $(loadavg_OBJECTS) $(loadavg_LDADD) $(LIBS)

make$(EXEEXT): $(make_OBJECTS) $(make_DEPENDENCIES) $(EXTRA_make_DEPEND
```

Above Large pieces of software can have fairly complicated makefiles. This is a portion of the makefile for Make itself

in rule definitions, but we need to use the function if we want to use it to set a variable. So $(wildcard *.c) will give us a list of all the source files in the directory. The second function is patsubst, which performs pattern substitution on strings. In this case, it is telling Make to replace each file name %.c with %.o, where % can be any string of characters. It's similar to the shell command

```
$ echo *.c | sed 's/\([^ ]*\)\.c/\1\.o/g'
```

Make recipes

As we mentioned above, Make recipes are shell commands that are inserted into the makefile to tell it how to build a particular rule.

Since recipes are intended to be passed to the shell, they are written in a different language from the rest of the makefile. Specifically, they are written in the usual shell-scripting language, rather than in the Make language. We have access to all the shell's capability within our recipes: all Make does is pass on the recipe to the shell for execution.

The most important exception to this rule is that recipes can contain Make variables. Above, we included the OBJECT_FILES variable inside the recipe for the player goal. A more accurate version of what we said above is: Make performs some cursory preprocessing of the recipe, by replacing variable references with the content of that variable.

Variable references in Make use the dollar sign $, which could cause a conflict if we want to include some shell variables in our recipe. In that case, we can escape the dollar sign by writing two of them together. For example, the C compiler on a system is often stored in the environment variable CC. In that case, we could modify the recipe for player to:

```
$${CC} -o player ${OBJECT_FILES}
```

We'll be looking at Make in more detail next issue.

Alex Cox
Part computing writer, part stay at home Dad, part breaker of things, Alex has been hacking his hardware since he first took his ZX Spectrum to bits.

Resources

- **An old Android phone**
  Check the wiki to see if it's supported https://wiki.lineageos.org/devices/
- **Android Developer Bridge**
- **For HTC phones, HTCDev registration**
  http://htcdev.com/bootloader
- **Google apps packages**
  http://opengapps.org
- **Pre-built ROM for building LineageOS**
  http://downloads.lineageos.org
- **SunShine for unlocking the bootloader**
  http://theroot.ninja
- **Official wiki**
  https://wiki.lineageos.org

Install LineageOS

Got an old, badly updated phone? Revive it with a fresh, clean, open source version of the Google mobile OS

Not for nothing do people often say that money changes everything. In the case of super-popular Android OS fork Cyanogenmod, money changed its parent company's philosophy, changed its direction from one of free software development to one of commercial software licensing, changed the company's CEO and project leader, and an eventual lack of money led to Cyanogen Inc. cutting back drastically on its operations and completely dropping support for Cyanogenmod. It also led to the community, which had been hugely involved in its development, snatching back its baby and rechristening it.

Thus we have the latest Cyanogenmod fork, and its de facto open-source successor, LineageOS – developed by that alienated community with large contributions from the original Cyanogenmod project leader, Steve Kondik. The name has changed – blame the corporate trademark machine – but the version number, 14.1 at Lineage's first release, follows directly on from that of Cyanogenmod, and the codebase has barely changed.

Before we explore the how, let's look at the why. LineageOS is perfect for those who want to take control of, and tinker with, their phones or tablets. It's not distinct from Android just because its devs felt like tinkering with Google's mobile OS; tangible improvements have been made to the OS which could mean you see better battery life, more private communication, and much less of the bloat that plagues most stock phone firmware. It's been developed by people who love their phones and want better from them.

Installing LineageOS over plain Android does have disadvantages, though. Google has recently implemented a system called SafetyNet on its Google Play store, which enables app developers to check if a device is in a 'known good' state. Lineage's developers, mindful of any backlash, have wisely chosen not to circumvent this check, which means you may find certain apps unavailable through traditional channels. There are generally workarounds for this – search the web for .apk files if you want to install items without using the Google Play store.

Get the prerequisites

To install LineageOS, you'll need a compatible phone or tablet, and the relevant cable to connect it to your PC. Check the full list at **https://wiki.lineageos.org/devices** – if you have a reasonably modern handset, you should be able to change your firmware without a problem, although don't expect to jump to a later version of Android than is supported by your device. Older devices are more or less left out in the cold, but you may be able to use the tips in this guide to install an old Cyanogenmod build.

Above LineageOS is especially effective on older devices that may no longer be receiving manufacturer updates

You'll also need ADB – the Android Developer Bridge – installed on your PC. See the Installing ADB box, above right, for directions; it's not particularly tricky. Naturally, you'll also need the precise version of LineageOS for your device. We're installing firmware here, meaning it deals with the specific hardware inside your device, and there's no catch-all version that fits every phone or tablet.

While it's possible to build your own custom package, and this is the best way to get the very latest features of LineageOS, you can simplify the process by downloading a pre-built ROM from **http://downloads.lineageos.org**.

Installing ADB

Android Debug Bridge is essential for deep-level communication with your phone's operating system, and fastboot deals with communicating with your phone's bootloader.

Head to a terminal window on a machine running a 64-bit OS and download the tool package containing both with `wget https://dl.google.com/android/repository/platform-tools-latest-linux.zip`, then unzip it with `unzip platform-tools-latest-linux.zip`. Now edit your bash user profile file with `sudo nano ~/.profile` and add the following to it to ensure the adb and fastboot commands are available from everywhere:

```
if [ -d "$HOME/platform-tools" ] ;
then
    export PATH="$HOME/platform-
tools:$PATH"
fi
```

Reboot your machine, and the Linux side of things should be done – now it's time for you to set up your smartphone or tablet.

On your device, open Settings > About and then dig down through its menus until you find the Android build number. Tap this seven times, then head back out to the main Settings screen to see a new entry near the bottom: Developer options. Select this, scroll down, and then switch on USB debugging. Plug your smartphone in to your PC and a connection should be established; type `adb devices` in a terminal window, and tap OK on your phone to allow the connection. If all has gone well, running `adb devices` again should list your smartphone as 'device' rather than 'unauthorised', and you can also run `adb shell` to pull up a limited Linux shell and control that device.

```
...
(bootloader)
(bootloader) < Please cut following message >
(bootloader) <<<< Identifier Token Start >>>>
(bootloader) 5543919FC6109DB8745D3BA655349D24
(bootloader) 01E00B7588F1D0BDF23059BB2B8B4237
(bootloader) EAFC62F327BF2AF8BF09E693A6FE479A
(bootloader) 84BCE848FEA2E94211601EDCDB11F463
(bootloader) 5803D7513C6F1C86919FD27F2F54A0CB
(bootloader) B21C60AE68EBCCC94935BBEEC74CD7BA
(bootloader) E5B85E24046BC2CB987EB136A1886CD6
(bootloader) A78E85DBD5B39FCD0536B4F022D207F3
(bootloader) 38F76D5942EF8CD2EBD60DBD128E6273
(bootloader) 2E8F9B94D96AA9D0C2588DA72695EBC4
(bootloader) 26EB89910C86A1430EF21A4F716773CC
(bootloader) 29B3CFE909817ADA47CE3D8940DC59E2
(bootloader) 25E42A345ED983C4286097C4D03073A2
(bootloader) 892A272967023E7537F22EEB1FA7641E
(bootloader) 37E7B1861359189ED538A086BC980AC4
(bootloader) F7392CA68FF8E1DCC51EF757480EF104
(bootloader) <<<<< Identifier Token End >>>>>
OKAY [  0.076s]
finished. total time: 0.076s
```

If you want to build your own, there's a guide on the Lineage wiki page for your specific hardware.

Finally, your device is going to need an unlocked bootloader. The specific directions for this are entirely dependent on the device you're looking to install to; we're using an HTC One M7 for this tutorial, and the guide will reflect that. Your device might already have its bootloader unlocked, and if it doesn't there's likely to be significantly different methodology for getting this done. Again, check the Lineage wiki for specific instructions, or try a tool like SunShine (**http://theroot.ninja**) as a last-resort effort if your device's manufacturer is being really stubborn.

Fastboot to freedom

Before we get into the meat of proceedings, let's reiterate that we're basing this guide on putting LineageOS onto an HTC One M7; your device may require fewer steps, or it may be vastly more complex. There are risks, too; see 'Proceed with care!' box, p38, to mitigate these.

First, we need to get the device into a vulnerable state – which means activating Fastboot mode. Fastboot is a protocol used to write data directly to your phone's flash memory, which is exactly where our custom firmware is going to reside.Things aren't quite as simple as that, of course, but let's reboot your smartphone into the relevant mode by typing **`adb reboot bootloader`** into a terminal. Once this is done, it should be pretty obvious from your phone's display, which will show you a wodge of information about the specific versions of specific things running on it, and say, somewhere, 'Fastboot USB'.

Now check that Linux can still see the phone by typing **`fastboot devices`** into a terminal. Depending on your setup, this might have to be done as a superuser; if so, instead run:

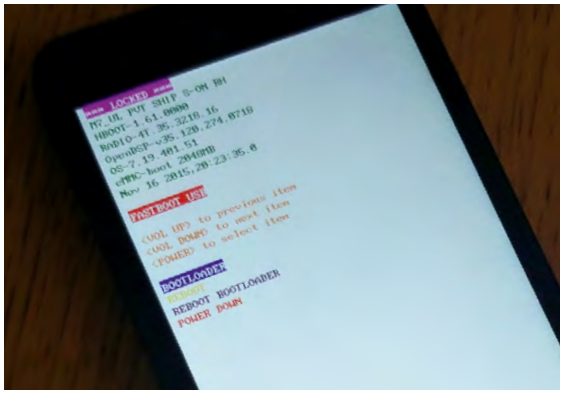
```
cd ~/platform-tools
sudo ./fastboot devices
```

Unlock your bootloader

One thing you'll probably notice about your device's screen is the word 'locked'. The vast majority of devices lock down their bootloaders – the code that runs when the device is first switched on – for obvious reasons; most manufacturers would rather the average user didn't cause them technical support issues or warranty disputes by installing their own custom ROMs. But by the same token, many manufacturers are happy to let developers do their own thing. Now that we're accessing fastboot, we can obtain our device's unlock token:

```
fastboot oem get_identifier_token
```

In our case, it's now off to the HTCDev website to register as a developer, followed by a visit to **http://htcdev.com/bootloader** to turn that token into a key. There's a fairly hefty warning shown on the site before we go through the process, as this really is the last chance to back out before your device is forcibly factory reset. Run through the instructions, copy your unlock token from the

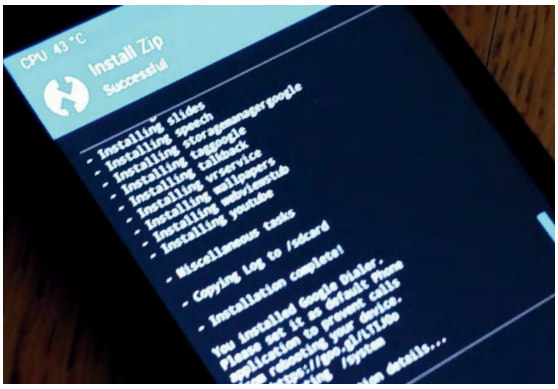


Left You're usually forced to control your device's fastboot screen using the power and volume buttons, since touchscreen drivers won't be loaded

Left If you run a terminal which allows corner-to-corner text selection, cutting out your device unlock token is trivial

Quick tip

ADB doesn't run much in the way of error correction, so if you're using a slightly flaky USB cable – or a USB extension lead – you may have issues transferring files. Keep it short, keep it clean.



Right A recovery loader, such as TWRP, accepts properly-constructed zip files and executes install scripts

terminal and paste it in the appropriate box, trimming off the '(bootloader)' or 'INFO' text to make it one clean block and leaving the start and end identifiers intact. If all has gone well, you'll be sent your device unlock key to your registered email address. Save this to the **~/platform-tools** directory, where your fastboot tool is located, then head back to your terminal, move to the **~/platform-tools** directory, and enter the following (replacing the filename if your device's manufacturer has sent you something different) to begin the unlock:

```
sudo ./fastboot flash unlocktoken Unlock_code.bin
```

You'll see yet another disclaimer, this time on your device's screen. This really is your last chance to back out, as your device will likely factory reset once you make the appropriate selection. Once this is complete head back to Settings and enable developer mode and USB debugging as before.

Recovery mode

Now that we've opened up access to your bootloader, it's time to trick your hardware into thinking something has gone disastrously wrong – a phone in recovery mode is one that's all too willing to accept a new, presumably working, firmware.

First, return to your device's bootloader by typing **adb reboot bootloader** and verify that its fastboot is now unlocked. Now point your web browser at **http://twrp.me**, and get the relevant version for your device. TWRP is a custom recovery package that overrides the one that comes preinstalled with your smartphone or tablet, which is essential as default recovery systems are usually locked to official firmware images. Copy the file to your **~/platform-tools** directory, open up your terminal in that directory, and then enter the following, switching the filename for that of your device:

```
sudo ./fastboot flash recovery twrp-3.1.1-0-m7.img
```

You now need to reboot your device and enter its recovery mode manually. On our test phone, we do this by holding the power and volume down buttons on boot; your method might be different. You'll see a screen very similar to the fastboot screen we've booted to previously; use the buttons stated on screen to select the Recovery option, and, once TWRP is loaded, use the 'Swipe to allow modifications' slider.

Flash your firmware

With TWRP active, check with **adb devices** that your computer can see your device – it should be listed in Recovery mode. Get the appropriate build of LineageOS for your phone or tablet from **http://download.lineageos.org**, and download the Google apps package (called gapps) from **http://opengapps.org** that matches your device's architecture, the version of LineageOS you're installing, and the number of in-built, stock apps you want to install. Pick Nano if you just want the essential elements, or Stock if you want the entire selection of apps that comes with the Nexus and Pixel lines. There are in-between options, too; click Variant at the top of the right column to see a comparison table.

Installing Google apps is optional, and we'll detail a completely open source alternative shortly, but you may find your Android experience quite limited without the likes of the Google Play store on board. It's worth noting that OpenGApps hosts its files on terrible servers, so you may need a little bit of time and perseverance to get them downloaded; we employed liberal use of **wget -c** to resume downloads that failed. Place the files in your **~/platform-tools** directory, and then upload them to your device with:

```
adb push lineage<filename>.zip /sdcard/
adb push open_gapps<filename>.zip /sdcard/
```

Note that we're uploading them to the **/sdcard/** directory on your device; this will be present even if it doesn't have a physical SD card slot, due to a quirk of Android's file layout. Essentially, **/sdcard/** is a shortcut for 'internal

Proceed with care!

What we're doing here is absolutely destructive. Flashing new firmware to your phone means your old firmware, the stock OS that came with your device and everything you've added to it since, will be erased, and while we'll take backups along the way there's no guarantee of them catching everything. If you're flashing your main phone, that means that your phonebook will be gone, your photos, everything. Should something go wrong during the process, there's even a possibility you'll brick your device. We're absolutely not trying to scare you away, but it's sensible to be aware of the realities, and sensible to take the right precautions.

At the very least, make sure that you've fully backed-up your device. Connecting it via USB will give you access to its internal storage, through which you should be able to manually back up your personal files. If you're signed in with a Google account, as you likely will be, you may also wish to consider trusting your data to its cloud services: Google Contacts can manage your phonebook, for example, and feed it back in to your device when you've installed your new firmware, and Google Photos stores an unlimited number of (reduced resolution) photos. You may as well employ belt as well as braces.

storage'; if you find that the **adb push** process doesn't work (and it's known to fail on certain devices) you may just be able to copy the files using your desktop file manager, particularly if it has found and mounted your hardware as a drive.

Back on your device, you can now take a backup of your internal storage as it is in its current state if you wish. This is usefyk if you ever want to return to stock firmware, as some manufacturers don't readily supply it. When you're happy to install, head to the Wipe menu and select Advanced Wipe. Tick the relevant boxes and slide the slider to completely clear your Cache, Data and System partitions, then go back to the main menu and select Install. Scroll through your internal storage, select the LineageOS zip file, and swipe the slider to flash your firmware. Be patient during this stage; it may look as if nothing is happening, but interrupting the process could mean your hardware ends up bricked.

Now, before rebooting your system repeat the same process with the Google apps file if you're using it. This has a slightly more lucid installer, so you'll see each app being installed as it goes on.

Gain root

Don't reboot quite yet. Advanced users may relish the chance to gain root access to their system. This isn't supplied by default with LineageOS, as it has obvious security implications – install a rogue app with root access, and you stand to have your device seriously compromised. It also isn't something everyone will need, but your hardware's potential definitely increases slightly with it active.

You can download LineageOS' superuser tool from **http://download.lineageos.org/extras**. Just grab the version that matches your OS, upload it to your phone as before, and flash it using TWRP's install tool; it'll take longer than you might expect for such a small file, but there are a lot of permissions to change on the device.

If you later decide to unroot your device, you'll find the relevant tool in the same place; just boot back into recovery, and flash it to unroot your device.

Initial set up

It is, at last, time to reboot your device. You'll be given the option to install the TWRP app on your system, which is entirely your decision; since you're now using a device with custom firmware it can make working with recovery a little easier, so it's not a bad idea. Now follows a very lengthy initial boot process, during which LineageOS churns through your device's storage making sure everything is set to go. Be patient, wait out the initial animation, and you'll pop into the initial user setup process. This is very closely related to that of stock Android, and gives you the chance to restore data from another device if you wish. You're also asked to sign in to Google services here; if you're not using Google apps or the Google Play store, feel free to skip this.

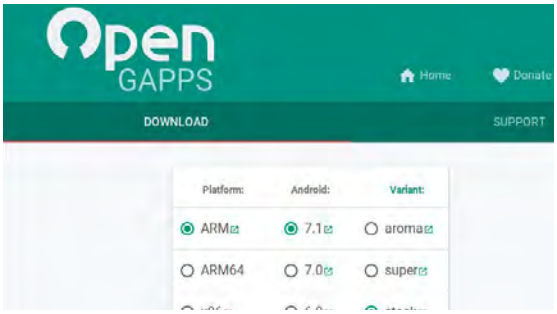
When you're through the Android part, you'll get a little LineageOS-specific configuration – we'd recommend switching on Privacy Guard, which is disabled by default, so you can control specifically what kinds of data the apps on your device are allowed to access. You can change your settings later in the Settings > Privacy menu, so if an app seems to be getting too big for its boots you can cut it down to size. Once that's done, you're in – with a clean, updated device, containing none of the manufacturer's original bloatware. If you didn't install Google Apps, you'll get a small stash of Lineage's open source alternatives to ensure your phone hardware maintains its required functionality.

Left If you use Lineage's superuser tool, consider removing it once you've done your privileged tasks – the removal tool is available from the same location

Get extras

If you've installed the root package, you'll now need to enable it. Head to the Settings app, and tap the Build number seven times to enable developer mode. Go back to the main settings screen, and enter the developer mode menu. We'd recommend switching on Advanced Restart at this point – this lets you quickly boot to recovery or your bootloader using the standard restart procedure – then scroll down to the root access option and choose the level of access you're willing to grant. The 'ADB' option will restrict root to commands solely sent through Android Developer Bridge, meaning you can gain root access (and a root shell) on your device, but deny applications the same rights.

There's a lot more control available in the developer options than you'll have previously seen in stock Android. You can really drill down to specifics, and enact some harsh policies over your hardware if you want to keep a close eye on things. Activating the 'Don't keep activities' option, for example, is perfect if you want to take a more paranoid hold of your apps, and switching on 'Enable local terminal' adds a new app that gives you shell access to your device as long as you've previously given root access to apps.



Left Not sure which Google apps to put on your 'new' phone? Download the Aroma variant, which offers an interactive installer to enable you to select them one-by-one

Quick tip

Fancy using Linux on your phone without any of this hassle? Grab 'Debian noroot' from the Google Play store – you'll then be able to run desktop apps on mobile devices.

Tam Hanna
is the CEO of consulting company Tamoggemon Holding k.s. He grew up under the influence of Eric Sink's classic essays on coupling and event orientated-programming.

Resources

- **Raspberry Pi 3**
- **PaHo documentation** http://bit.ly/lud_paho
- **Explanation of package flows in MQTT QoS scenarios** http://bit.ly/lud_mqtt1 http://bit.ly/lud_mqtt2

PART TWO

Deploy MQTT on a Raspberry Pi running Android Things

Android Things permits the fusing of rich visualisation and hardware access: an ideal tool for the Internet of Things

In part one of this series [Tutorials, p42, LU&D 186] we used desktop-based examples of MQTT to avoid the complexity of deploying it to mobile devices. But MQTT is not a desktop-centric protocol, and given that you are now somewhat seasoned in its use, we can move forward and look at how to use it on a mobile device.

People now expect all kinds of graphics and one exciting way to handle this problem is Android Things; it's a slimmed-down version of Android optimised for process computers such as the Raspberry Pi. Developers deploying it can use existing Android classes for visualisation, while interacting with non-real-time hardware. The following steps use a Raspberry Pi 3 running a recent version of Android Things.

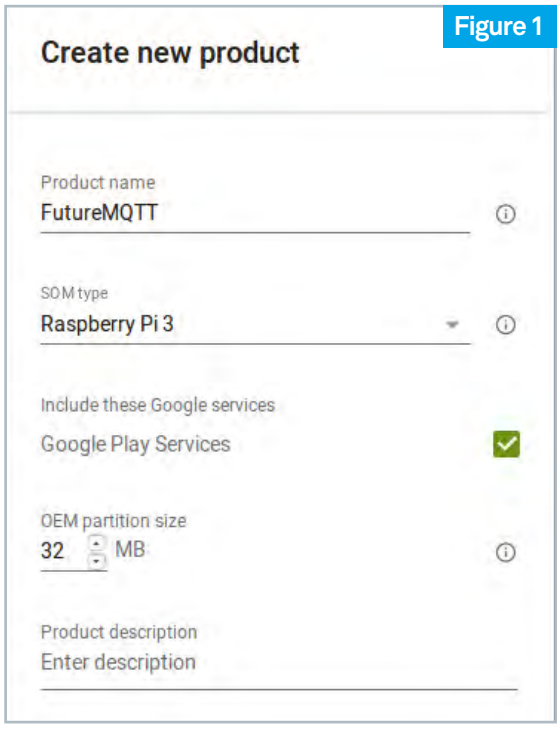
Because of the wide-reaching communality between Android and Android Things, examples can also be used on a phone skip any parts related to the process computer and run the code on your phone of choice if you don't feel like handling the additional hardware burden.

When Android Things was introduced, developers could download starter images from Google's web site. Google intends Android Things to be part of its IoT ecosystem; for you, that means the components must now be downloaded from the Console which is available at https://partner.android.com/things/console/?pli=1. Using it requires a Google account, but does not – at time of press – require a paid Play Store subscription.

After logging into the console, click the 'Create New Product' button and configure the various settings according to **Figure 1**. As of this writing, Android Things doesn't fit onto a 4GB card, so it's a good idea to configure the project's OEM partition with ample space for growth. Next, switch to the Factory Images tab and click the 'Create Build Configuration' button. Scroll down to the build configuration and click the download link to start the deployment of an image and install it onto an SD card of choice as you would with a normal Raspbian image. Choosing the right SD card is a science of its own – we have a set of 32GB cards on hand and simply use one of these during debugging. In theory, you could use the Pi's Wi-Fi transmitter. From experience, though, this is an extremely bad idea: debugging is a resource-intensive operation, which is greatly affected by latency. We suggest using an Ethernet cable to connect the process computer to your workstation. Then, make sure that the network settings look as shown in **Figure 3** (p43).

Android Things is not good at detecting hardware which has been connected after power-up. Connect a mouse,

Above Fun fact: Andy Stanford-Clark, co-developer of MQTT, was a home-automation pioneer. To fix a serious mouse problem in his attic, he used MQTT to build a multi-mouse trap so he didn't have to keep going up in the roof to check if the traps had gone off or not



Left These settings create an image which can be run on the Raspberry Pi

the network cable, the HDMI screen and power up. Don't worry if the first start takes up to half an hour: the stock image must first expand its partition table. When done, Android Things will display a stock startup screen.

Developers switching to Android Things from a phone might wonder where the program starter is. Well, it doesn't exist. An Android Things device is a one-trick pony which is not intended to give the user any kind of choice over the programs run on it.

Make it compile

Any developer working with Android Things is strongly recommended to use Android Studio 3.0 – the latest version of the IDE contains a variety of features related to IoT. For example, the project creation assistant can be used to directly create a project skeleton containing the various Android Things expansions by selecting the option shown in **Figure 2** (see p42). Next, make sure to add an activity – we need to display some information on the screen during program execution.

Making the Raspberry Pi visible to Android Studio is accomplished via the ADP bridge. It can also be used in networked mode via the `connect` command:

> Android Things is a slimmed-down version of Android optimised for process computers such as the Raspberry Pi

```
tamhan@TAMHAN14:~/Android/Sdk/platform-tools$
./adb connect 10.42.0.44
connected to 10.42.0.44:5555
```

MQTT support for Java applications usually comes in the shape of Eclipse PAHO, a complete event-driven MQTT library. Thanks to Android Studio's use of the Gradle build system, deploying the product is simple.

The Build scripts folder contains a group of build files; the one marked Project governs the entire solution, while the one marked Module governs the application. First, open the file responsible for the whole solution and modify the repositories block to add a reference to a server provided by the foundation:

```
buildscript {

    repositories {
        google()
        jcenter()
        maven {
            url "https://repo.eclipse.org/
content/repositories/paho-snapshots/"
        }
    }
```

With that out of the way, go to the build.gradle. There, the dependencies block must also be modified so that PAHO becomes available:

```
dependencies {
    implementation fileTree(dir: 'libs',
include: ['*.jar'])
    compile 'org.eclipse.paho:org.eclipse.
paho.client.mqttv3:+'
    compile 'org.eclipse.paho:org.eclipse.
paho.android.service:+'
    compile "com.android.support:
support-v4:+"
```

Having saved one of the files, Android Studio will display a yellow banner at the top of the screen. Click the 'Sync Now' hyperlink to start the Gradle build process – missing libraries will be downloaded automatically if an internet connection is available.

Before starting to work with the MQTT client, let's spend a bit of time thinking about what can happen during message delivery. Reliable package delivery is not a given the moment you leave the desktop. Messages over GSM can both disappear and be delivered more than once. The severity of this depends on the application and how critical it is. In the case of MQTT, three different priority levels are defined. Level zero (QoS0) means zero effort: the sender will take a stab at sending the message and stop. While the underlying TCP protocol will invest some effort to make sure that the message arrives, no guarantee is given. However, the lack of retransmission logic ensures that the message will be sent and (usually) received a maximum of once. In practice, QoS0 is ideally suited to situations where 'random' information must

Be aware of overloads!

Creating images is done exclusively on Google's servers, which can run out of processing power from time to time. Due to that, don't be surprised if the compilation takes up to 15 minutes; fortunately, a progress bar allows you to keep tabs on the process as it runs.

Feeble connection

Developers who usually connect their workstation to the target device using USB will be shocked to find that debugging via a network is labour-intensive. The most important obstacle is ADB: it does nothing to re-establish a lost connection to a target. If your workstation hibernates or you disconnect the network cable, be prepared to return to the console to re-enter the connect command.

be transferred, such as status data which is not of great significance to the system. Sending a message with a level of 1 (QoS1) ensures that the message will be delivered at least one time, and it is possible that the message will be received multiple times. This is accomplished by the use of the PUBACK message: it gets sent in response to a published message, and tells the client to stop further transmission attempts. This message, however, doesn't have to arrive, which would lead to repeated transmissions. Setting a QoS of 2 requires the highest amount of processing, but ensures that the message gets delivered exactly once. This is accomplished by a complex process: in addition to the exchange introduced in QoS1, the client must accomplish an additional exchange to send the actual message off.

Another very interesting problem in the domain of MQTT arises if the client sends a message with one quality of service level, while the recipient has signed up to the broker using another one. In this case, the QoS level used by the sender determines the maximum quality possible, while the actual delivery to the individual client is accomplished using the QoS level required by their registration.

Make it connect

The manifest of our Android application requires a slew of changes. First, a group of permissions must be declared so that the program can interact with the internet and other sensitive parts of the operating system. Secondarily, we also declare a service: this is a special part of the operating system which allows us to perform all kinds of operations in the background:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.
android.com/apk/res/android"
      package="com.tamoggemon.futuremqtt">
      <uses-permission android:name="android.
permission.WAKE_LOCK" />
      <uses-permission android:name="android.
permission.WRITE_EXTERNAL_STORAGE" />
      <uses-permission android:name="android.
permission.ACCESS_NETWORK_STATE" />
      <uses-permission android:name="android.
permission.READ_EXTERNAL_STORAGE" />
      <uses-permission android:name="android.
permission.INTERNET" />
      <application>
          <uses-library android:name="com.
google.android.things" />
          <service android:name="org.eclipse.
paho.android.service.MqttService"/>
```

Developers unfamiliar with Android can rest easy here – we don't need to implement the service by hand. Instead, simply including it via a service tag is enough, because the PAHO team has done most of the work for us.

With that, create a member variable of the type `MqttAndroidClient` and get connected to the server. That member variable must then be populated:

```
@Override
protected void onCreate(Bundle
savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      myClient=new MqttAndroidClient(getApplica
tionContext(),"tcp://10.42.0.1:1883","TamsClient");
```

The constructor does not accept a second parameter for the port; instead, port information is encoded into the URL. It is then parsed during program execution, which – by and large – leads to comparable results. In the next step, the actual configuration takes place. Enable the Automatic Reconnect feature, as it lets us survive a network outage:

```
      MqttConnectOptions mqttConnectOptions =
new MqttConnectOptions();
      mqttConnectOptions.
setAutomaticReconnect(true);
      mqttConnectOptions.setCleanSession(false);
    try {
          myClient.connect(mqttConnectOptions,
null, new IMqttActionListener() {
              @Override
            public void onSuccess(IMqttTokena
syncActionToken) {
                  DisconnectedBufferOptions
disconnectedBufferOptions = new
DisconnectedBufferOptions();
                  disconnectedBufferOptions.
setBufferEnabled(true);
                  disconnectedBufferOptions.
setBufferSize(100);
                  disconnectedBufferOptions.
setPersistBuffer(false);
                disconnectedBufferOptions.
setDeleteOldestMessages(false);
                  myClient.setBufferOpts(dis
connectedBufferOptions);
                  Log.e("Future", "Link
up!");
              }
              @Override
```

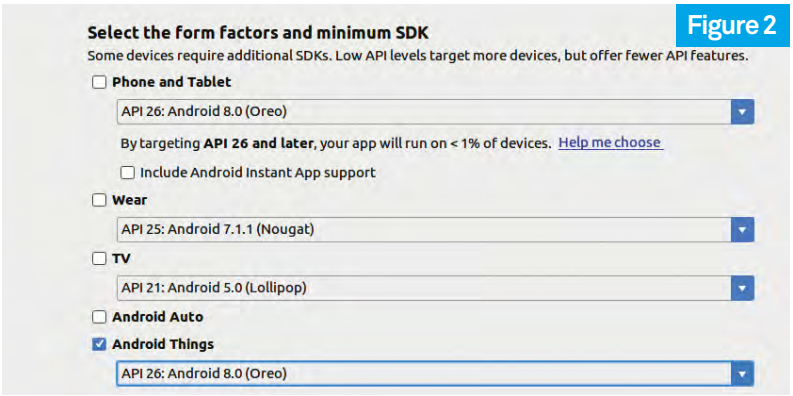

Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.
Phone and Tablet
API 26: Android 8.0 (Oreo)
By targeting API 26 and later, your app will run on < 1% of devices. Help me choose
Include Android Instant App support
Wear
API 25: Android 7.1.1 (Nougat)
TV
API 21: Android 5.0 (Lollipop)
Android Auto
Android Things
API 26: Android 8.0 (Oreo)

Figure 2

Below This tick box saves you a *lot* of hassle...

```
                public void
onFailure(IMqttToken asyncActionToken, Throwable
exception) {
                    Log.e("Future", "Link
fail!");
                }
            });
```

The second part of our program is focused on establishing the connection. The `connect` method takes an connection object property and an event listener which gets invoked during the link-up process. For us, the `onSuccess` method is especially interesting as it is responsible for setting up the caching object. It is an instance of the type `DisconnectedBufferOptions`, and handles caching while no connectivity to the server.

Check once again that the IP address is correct, and run the program. If everything is correct, the Pi will connect to the Mosquitto server on your workstation and emit a success report into the debugger console.

Now that the basic link-up issue is sorted, the next problem is establishing message transfers. Open the MainActivity layout, and a total of three buttons intended for transmission of messages at different QoS levels.

Given that we have a field of buttons mostly doing the same thing, it is now time to look at the old ways of Android programming. `View.OnClickListener` instances don't necessarily have to be created as a member variable; it's also possible to implement one listener in the activity, and to take apart the events by using the view parameter indicating the object responsible for admission. This way, the setup in the constructor becomes short and sweet:

```
cmdQos0=(Button)findViewById(R.
id.cmdSendMessageQos);
cmdQos0.setOnClickListener(this);
cmdQos1=(Button)findViewById(R.
id.cmdSendMessageQos1);
cmdQos1.setOnClickListener(this);
cmdQos2=(Button)findViewById(R.
id.cmdSendMessageQos2);
cmdQos2.setOnClickListener(this);
```

Actually sending out the messages is not difficult. Our handler first determines the quality of service level by analysing the button, and then goes through the procedures required by the Java language standard:

```
@Override
```

> "In theory, you could use the Pi's Wi-Fi transmitter. From experience, though, this is an extremely bad idea when debugging"

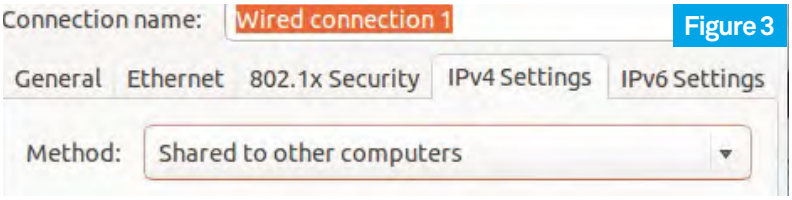


Figure 3

Above An Ubuntu workstation can also act as a DHCP server

```
public void onClick(View view) {
      int myQos=0;
      if(view==cmdQos0)myQos=0;
      if(view==cmdQos1)myQos=1;
      if(view==cmdQos2)myQos=2;
      try {
          MqttMessage message = new
MqttMessage();
          message.setQos(myQos);
          message.setPayload("Hello Raspi".
getBytes());
          myClient.publish("tamstest",
message);
      } catch (MqttException e) {
          System.err.println("Error Publishing:
" + e.getMessage());
          e.printStackTrace();
      }
}
```

Click the 'Debug' button to chase the next version of the example to the Raspberry Pi.

For the first test, click one of the buttons and use the MQTT listener in your workstation's command line – the message will show up normally. You can, of course, also enlist the services of our Qt program from the last issue.

The next step involves testing the quality-of-service feature. For this, disconnect the network cable from the Pi, then click the button for quality-of-service level 0 a few times. Finally, reconnect the network cable; the command line will not be populated with any additional events. You can verify that the connection was established successfully by hitting the quality-of-service 0 button a few more times after plugging in the network cable – once the process computer has received its IP address, message delivery will continue. However, the messages queued during the offline period will not be retransmitted. If another QoS button is used during the offline phase, the message will show up after the network cable is reconnected.

Keep in mind that this is not an instantaneous process. PAHO implements a stepped back-off approach: if no connection is established, a one-second waiting time is planned. If, after that time, the connection still fails, that time is doubled to two seconds. This process repeats until a back-off time of two minutes is reached, which is then considered the maximum. Don't be surprised then, if it takes a few seconds until the messages show up in the command line after the network cable is reconnected.

While the Raspberry Pi is a nice embedded system, it is neither the cheapest nor the smallest – so the next part of this feature will introduce you to the ESP32!

Toni Castillo Girona
holds a degree in Software Engineering and an MSc in Computer Security and works as an ICT research support expert in a public university in Catalonia (Spain). Read his blog at **http://disbauxes.upc.es**.

Resources

- **The Penetration Testing Execution Standard** http://bit.ly/pen_test
- **Exploit Database** https://www.exploit-db.com
- **Sift** http://bit.ly/lud_sift
- **Nmap Script Engine** http://bit.ly/lud_nmap
- **Shell-storm** http://shell-storm.org/shellcode
- **OWASP ZSC** http://bit.ly/lud_zsc
- **Metasploit Framework** https://metasploit.com

Use reliable exploits from trusted sources in your code

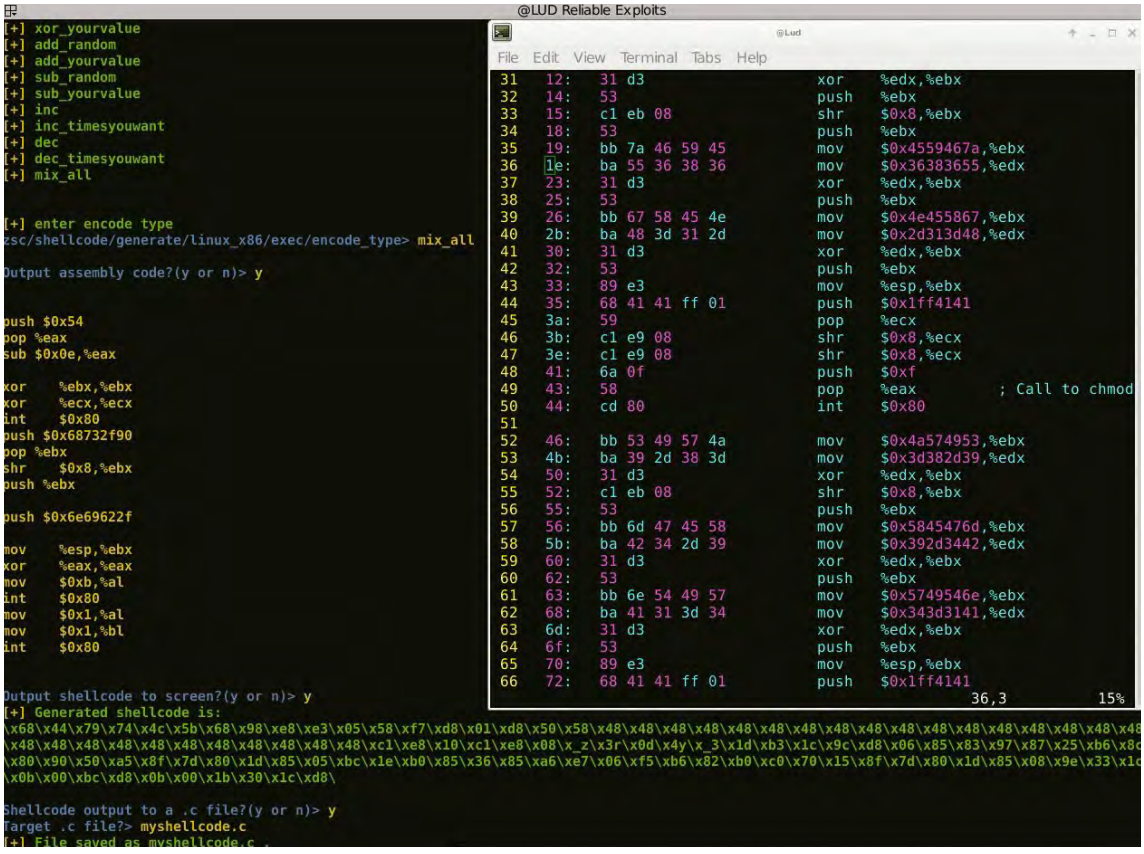
Learn how to find, modify and use exploits which are regarded as reliable in your penetration testing

Are there such things as reliable exploits? Well, if you're acquainted with the traditional seven-stage penetration-testing engagement definition, then you probably know that performing the exploitation and post-exploitation stages is like juggling on quicksand. Not that detecting the vulnerability in the first place is a bulletproof process anyway, but of course at some point you will need to execute some exploits. Even if you code your own, some system crash may eventually hit you right in the face, hard.

So is there any hope? For those who test their exploits over and over and over again, there may be. For those who use well-designed exploitation frameworks with verified exploits, there may be. For those who analyse what a particular type of shell-code within an exploit really does, there may be. And, of course, for those who are about to read and follow this tutorial, there may be. Mind you, this is not like maths: myriads of little things can go wrong at any given point in time. But as long as you look for verified exploits while staying away from unreliable sources and Proof of Concept code (PoCs), you should be (almost) okay.

The Exploit Database

The Exploit Database (E-DB, see Resources), maintained by Offensive Security – the same guys who birthed Kali Linux – is an amazing source of exploits and shell-code. It also hosts the Google Hacking Database (GHDB). E-DB is extremely easy to navigate, whether online or by using their `searchsploit` script. You can keep a local, offline


@LUD Reliable Exploits
[+] xor_yourvalue
[+] add_random
[+] add_yourvalue
[+] sub_random
[+] sub_yourvalue
[+] inc
[+] inc_timesyouwant
[+] dec
[+] dec_timesyouwant
[+] mix_all
[+] enter encode type
zsc/shellcode/generate/linux_x86/exec/encode_type> mix_all
Output assembly code?(y or n)> y
push $0x54
pop %eax
sub $0x0e,%eax
xor %ebx,%ebx
xor %ecx,%ecx
int $0x80
push $0x68732f90
pop %ebx
shr $0x8,%ebx
push %ebx
push $0x6e69622f
mov %esp,%ebx
xor %eax,%eax
mov $0xb,%al
int $0x80
mov $0x1,%al
mov $0x1,%bl
int $0x80
Output shellcode to screen?(y or n)> y
[+] Generated shellcode is:
Shellcode output to a .c file?(y or n)> y
Target .c file?> myshellcode.c
[+] File saved as myshellcode.c .
File Edit View Terminal Tabs Help
31 12: 31 d3 xor %edx,%ebx
32 14: 53 push %ebx
33 15: c1 eb 08 shr $0x8,%ebx
34 18: 53 push %ebx
35 19: bb 7a 46 59 45 mov $0x4559467a,%ebx
36 1e: ba 55 36 38 36 mov $0x36383655,%edx
37 23: 31 d3 xor %edx,%ebx
38 25: 53 push %ebx
39 26: bb 67 58 45 4e mov $0x4e455867,%ebx
40 2b: ba 48 3d 31 2d mov $0x2d313d48,%edx
41 30: 31 d3 xor %edx,%ebx
42 32: 53 push %ebx
43 33: 89 e3 mov %esp,%ebx
44 35: 68 41 41 ff 01 push $0x1ff4141
45 3a: 59 pop %ecx
46 3b: c1 e9 08 shr $0x8,%ecx
47 3e: c1 e9 08 shr $0x8,%ecx
48 41: 6a 0f push $0xf
49 43: 58 pop %eax ; Call to chmod
50 44: cd 80 int $0x80
51
52 46: bb 53 49 57 4a mov $0x4a574953,%ebx
53 4b: ba 39 2d 38 3d mov $0x3d382d39,%edx
54 50: 31 d3 xor %edx,%ebx
55 52: c1 eb 08 shr $0x8,%ebx
56 55: 53 push %ebx
57 56: bb 6d 47 45 58 mov $0x5845476d,%ebx
58 5b: ba 42 34 2d 39 mov $0x392d3442,%edx
59 60: 31 d3 xor %edx,%ebx
60 62: 53 push %ebx
61 63: bb 6e 54 49 57 mov $0x5749546e,%ebx
62 68: ba 41 31 3d 34 mov $0x343d3141,%edx
63 6d: 31 d3 xor %edx,%ebx
64 6f: 53 push %ebx
65 70: 89 e3 mov %esp,%ebx
66 72: 68 41 41 ff 01 push $0x1ff4141
36,3 15%

Above Aren't you tired of generating shell-code with msfvenom? We are, full stop

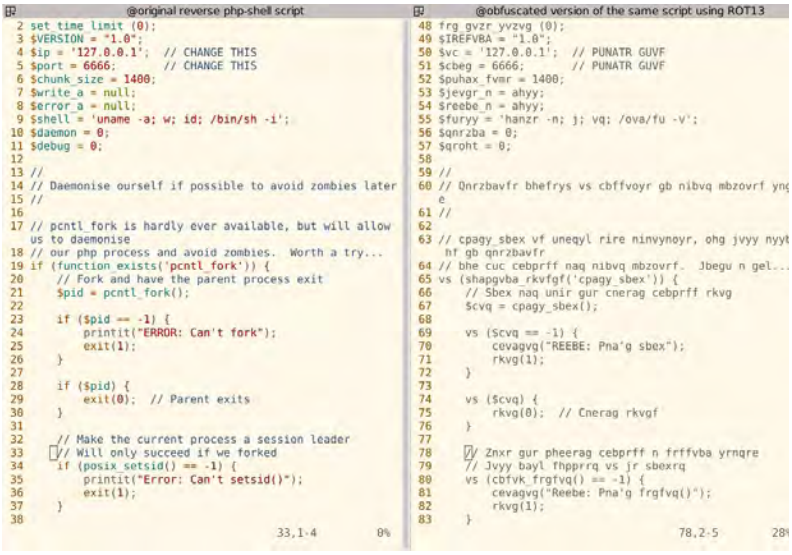
copy of the whole database if you want and update it as needed. One standout feature of E-DB is the huge amount of available exploits and shell-code that have been tested and verified by Offensive Security staff. A verified exploit does not necessarily mean that it is 100 per cent safe to use; it only means that it has been tested, and it delivers (without crashing the target or harming your own computer, that is). If you're not a fan of integrated exploitation frameworks such as Metasploit (see Resources), then you should look for exploits here first. Clone E-DB from github: `git clone https://github.com/offensive-security/exploit-database.git /opt/exploit-database`. Then, make a link to the `searchsploit` script: `ln -s /opt/exploit-database/searchsploit /usr/local/bin/searchsploit`. Now you can search for exploits or shell-code locally. Let's imagine you are interested in exploits for Android that are neither PoCs nor DoS; use the `-t` and `-exclude` flags for this:

```
searchsploit -t android --exclude="PoC|/dos/"
```

You can also look for exploits that are present in the Metasploit framework:

```
searchsploit -t "(metasploit)" -o --exclude="PoC|/dos/"
```

Once you have found the right exploit, mirror it: `searchsploit -m EDB-ID`. The EDB-ID is the number given to the exploit (you can see it in the Path column). For example, let's imagine you are interested in Stack Clash Linux exploits, run: `searchsploit -t "Stack Clash Linux" -o -exclude="PoC|/dos/"`. As of writing, there are four exploits. You want the first one, 42775.c. Mirror it now: `searchsploit -m 42275`. This will copy the exploit to your current working directory. Of course you can also download any exploit from the E-DB website by using `wget` or `curl`. Type this in your CLI if you want to download the previous exploit: `wget https://www.exploit-db.com/raw/42275/ -O 42275.c`. One annoying thing about E-DB, however, is that the exploits are formatted using DOS carriage-return (\r) and new-line (\n) line endings,so you might want to convert the exploits to UNIX format before using them: `dos2unix EXPLOIT_FILE`. You'll probably be interested in checking whether a particular exploit or shell-code has been verified or not. In your browser, navigate to E-DB/Exploits/Remote Exploits; you will see a column labelled 'V'. If a particular exploit has been tested and verified, there's a tick icon next to it. When the exploit has not (yet) been tested, there's a clock icon instead. So modifying the `searchsploit` bash script in order to check whether a particular EDB-ID has been verified or not is easy; we've done that already. Get this version from the coverdisc, but before executing it type: `cat /opt/exploit-database/files_shellcodes.csv /opt/exploit-database/files_exploits.csv > /opt/exploit-database/files.csv`. You'll need curl (`#apt-get install curl`) and sift (see Resources). Copy the sift binary to `/usr/local/bin` and put the modified `searchsploit` script somewhere in your home directory. Now you can determine if a particular exploit has been verified by using the `-V` flag, like this:



Above Sometimes a canonical example of weak encryption is all you need to obfuscate a script

```
./searchsploit -V 42762
Verified: NOT YET
```

A verified exploit or shell-code will be reported as 'Verified: OK'. According to E-DB, its exploit database is updated daily, so you can keep your local copy up-to-date too by running: `sudo searchsploit -u`.

> "The Exploit Database (E-DB), maintained by Offensive Security, is an amazing source of exploits and code"

Nmap and its script engine

Nmap is more than just a port scanner. Thanks to its script engine, you can perform exploitation as well during your pen-testing engagements. According to the nmap documentation, its scripts are classified in different categories. As far as exploitation is concerned, you will be interested in the 'exploit' category. Install nmap on your computer: `apt-get install nmap`. Then get a list of all the available exploits by running:

```
nmap -script-help exploit
```

Some of these scripts are far from safe. Those tagged as 'Intrusive' may harm the target, and while they may not necessarily crash the remote system, the risks are just too high to dismiss. Therefore, you probably want to stay with those not tagged as Intrusive. Nmap allows you to

Metasploit module development

For those cases where you need to alter an existing exploit or code a new one within Metasploit framework, dealing with its module construction is a must. Luckily for you, it's only Ruby. Thanks to Metasploit mixins, coding new modules or customizing existing ones is not that hard. Metasploit handles most of the hard stuff itself. See **https://www.offensive-security.com/metasploit-unleashed/building-module**.

Take the challenge!

Using OWASP ZSC, download the exploit with ID=887 to disk. This is an obfuscated shell-code that creates a new user ALI with password ALI when executed. Locate the line that gets written to **/etc/passwd**. Change the user name ALI to the user name LUD in the shell-code and execute it for real (on a VM, mind you!). Stuck? Get tcg_exploits_187_solution.pdf from the coverdisc.

use logical operators to narrow the search. So if you want to get a list of all the non-intrusive exploits, run:

```
nmap -script-help "not intrusive and exploit"
```

New scripts are added to Nmap from time to time. Make sure to have your local scripts repository up-to-date by running `nmap -script-updatedb`.

Using Metasploit

Metasploit is a powerful exploitation framework that has been around for quite some time. It ships out of the box with some GNU/Linux pen-testing distros (Kali, Parrot and so on), and it's heavily used by pen-testers and security professionals alike – we'll cover Metasploit in a tutorial soon. This framework is great because it includes not only exploits, but some additional tools such as msfvenom to generate obfuscated shell-code. When it comes to reliable exploits, the Metasploit creators (Rapid7) keep an online database with all the available exploit modules (**https://www.rapid7.com/db/modules**). For every single exploit, a Reliability value tells you how reliable this particular exploit is. The Rapid7 website defines the ExcellentRanking value as "No typical memory corruption exploits should be given this ranking unless there are extraordinary circumstances". Indeed, if you think about abusing buffer overflows with ROP Gadgets, you would expect reliability to be far from perfect. Anyway, you can either look for a particular exploit using the Rapid7 website, or locally using `msfconsole`. But of course you'll probably want to look for reliable (so to speak) exploits. Metasploit includes an utility to do so: `/usr/share/metasploit-framework/tools/modules/module_rank.rb`. If you have a quick look at this Ruby script, you will see the Ranking values associated with their corresponding string names, for example Excellent=600, Great=500, and so on. Now, let's look for Linux exploits with a minimum rank of Great:

> "Using the term 'reliable' with exploits that abuse a memory corruption flaw is a bit daring, to say the least"

```
./module_rank.rb -f Exploit -m Great |egrep
--color=yes "exploit/linux"
```

If you are using Metasploit on a GNU/Linux Kali distro, you can update the framework and its modules using `apt` like this: `apt update; apt install metasploit-framework`. Otherwise, run `msfupdate`.

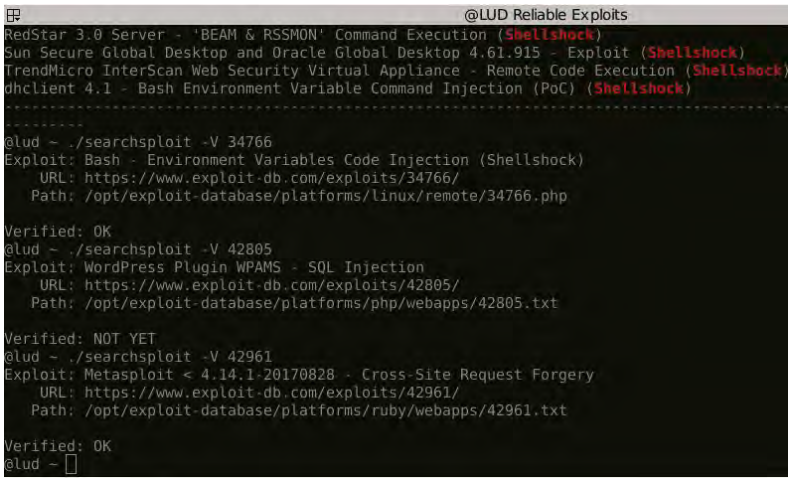
Reliable shell-code

Some exploits may contain shell-code whereas some others may not; as you have already seen, using the term 'reliable' with exploits that abuse a memory corruption flaw is a bit daring, to say the least. Of course, shell-code in itself can be reliable or, at least, as reliable as any other piece of code you may think of. So an exploit that uses some shell-code must ensure that this shell-code is reliable as well. Sometimes you will need shell-code for your own purposes, or for some new exploit you are working on. It is really easy to generate new shell-code and obfuscate it with OWASP ZSC (`msfvenom` from Metasploit framework can do the same thing, but its obfuscating techniques are now widely known and most antivirus and security software are no longer fooled), but for the time being let's focus on reliable sources for already generated shell-code. One such reliable source is, as you may have guessed, E-DB. For example, you can search for Linux x86 shell-code that spawns a new shell by running:

```
./searchsploit exec bash "lin_x86/shellcode"
```

Once you have found the shell-code that may suit you, don't forget to determine if it has been verified or not: `searchsploit -V EDB-ID`.

Another good source of shell-code is Shell-storm (see Resources). Shell-storm has a very straightforward API that enables you to create your own scripts, if need be, to query its database. OWASP ZSC makes use of this API (see Resources). According to the Shell-storm website it has stopped accepting new shell-code, but this database can still be useful, though. You can install OWASP ZSC in order to search and download shell-code from Shell-Storm. First clone its repository: `git clone https://github.com/zscproject/OWASP-ZSC`. Then install it: `cd OWASP-ZSC; python installer.py`. Execute the tool with `zsc` and then type the following commands, one after the other, each one followed by Return: `shellcode/search/add new root`. Once you have spotted the one you want, execute the following commands to download it: `shellcode/download/shellcode_id`, where `shellcode_id` is the ID of the shell-code you want to download. If you don't like ZSC CLI, use a one-liner shell command:


@LUD Reliable Exploits
RedStar 3.0 Server - 'BEAM & RSSMON' Command Execution (Shellshock)
Sun Secure Global Desktop and Oracle Global Desktop 4.61.915 - Exploit (Shellshock)
TrendMicro InterScan Web Security Virtual Appliance - Remote Code Execution (Shellshock)
dhclient 4.1 - Bash Environment Variable Command Injection (PoC) (Shellshock)
@lud ~ ./searchsploit -V 34766
Exploit: Bash - Environment Variables Code Injection (Shellshock)
URL: https://www.exploit-db.com/exploits/34766/
Path: /opt/exploit-database/platforms/linux/remote/34766.php
Verified: OK
@lud ~ ./searchsploit -V 42805
Exploit: WordPress Plugin WPAMS - SQL Injection
URL: https://www.exploit-db.com/exploits/42805/
Path: /opt/exploit-database/platforms/php/webapps/42805.txt
Verified: NOT YET
@lud ~ ./searchsploit -V 42961
Exploit: Metasploit < 4.14.1-20170828 - Cross-Site Request Forgery
URL: https://www.exploit-db.com/exploits/42961/
Path: /opt/exploit-database/platforms/ruby/webapps/42961.txt
Verified: OK
@lud ~

Below Oh yes, Metasploit has security flaws too. Head for the hills!

```
zsc -s download shellcode_id -o outputfile.c
```

The Metasploit framework ships with a good deal of shell-code too. You can search for 'reliable' shell-code in Metasploit by using module_rank.rb, as previously mentioned. This time use `-f Payload` instead of `-f Exploit`. Let's imagine you are looking for Linux x86 exploits with a ranking of Normal:

```
./module_rank.rb -m Normal -f Payload|egrep --color=yes "linux/x86"
```

Now, perform a module enumeration with `Ranking={Good,Great,Excellent}`.

Test your shell-code

If the exploit you are about to use does not come from a reliable source, ensure you understand what it really does and test it before using it against a real target. Moreover, if the exploit does contain shell-code, you can either replace it with shell-code generated by yourself (by any feasible means, say `msfvenom`, ZSC, E-DB, or by coding it yourself), instead of using it blindly, or you can analyse the shell-code to determine whether it is safe and does what it claims to do. Testing shell-code is not really that hard because it tends to be relatively small in size – just a bunch of assembly instructions and that's all. Of course it can be obfuscated (in fact, it should be obfuscated!), which renders the process of its analysis a bit more difficult. Any encoding scheme used for any particular shell-code can be defeated by means of emulation [see Tutorials, p44, **LU&D** 185]. Whenever dealing with unreliable shell-code, proceed as follows:

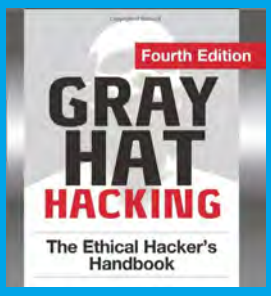
> "If the exploit you are about to use does not come from a reliable source, ensure you understand what it really does"

1. Generate a valid C source file containing the shell-code (like the files generated by, say, ZSC). Then, use any debugger (gdb or r2) to debug the code. This must be performed on a virtual machine if you want to actually let the code execute any syscall (like creating files, opening sockets and so on).

2. Emulate the code with ESIL or Angr. You do not need to emulate everything to understand what the code is doing. You can pick those parts within the code that seem complex or obfuscated. It always pays to understand what a particular shell-code does in order to change its behaviour. Of course you can use shell-code generators too. For example, within E-DB you can look for shell-code generators by executing: `searchsploit "linux/x86 generator/"`. Pick 13364 (make sure it has been verified first: `searchsploit -V 13364`) by using `searchsploit -m 13364`. Edit 13364.c and change the parameters for the functions SET_PORT and SET_IP to something else, save the file and compile it with: `gcc -g -O0 -m32 -z execstack -fno-stack-protector 13364.c -o 13364`. Next, open it within a gdb session: `gdb -q ./ 13364`. Set a break-point on line 131 of 13364.c: `b 13364.c:131` (where the call to the shell-code is made). Then run the program: `r`. The execution will break at `__asm__("call sc")`. If you now disassemble the shell-code (variable sc) within GDB, you will get the new settings (IP and PORT): `disas sc`. Take note where the OFFSET for 'sc' is and where the last shell-code instruction OFFSET is (right after the int $0x80 instruction). Then, dump it to a file with: `dump binary memory shellcode.bin 0x56557040 0x56557092` (where the first OFFSET is the address pointed to by `sc -&sc-` and the last OFFSET is the address right after the instruction int $0x80). Now, use `xxd` to generate the shell-code in C format: `xxd -i shellcode.bin`. Voila! You now have the shell-code with your own PORT and IP address. Of course you can also use OWASP ZSC or Metasploit (`msfvenom` or `msfconsole`) to generate shell-code without that hassle. Don't use obfuscated scripts unless you are able to de-obfuscate them first before executing them. It is always better to download a reliable script from reliable sources and then obfuscate it (or code your own). OWASP ZSC can do that for you too; let's play with a PHP reverse shell. Get php-reverse-shell-1.0.tar.gz from `http://bit.ly/lud_reverse`; untar it and then obfuscate it by running the following commands within ZSC: `obfuscate/php/php-reverse-shell.php/rot13`. It may ask you to remove the `<?php, <?` tags; type `yes`. Open the PHP script – it will look obfuscated (pictured, p45).

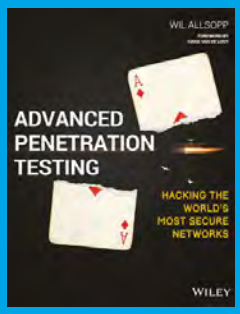
WHAT NEXT?

Read these essential books about pen-testing



1 Gray Hat Hacking, Fourth Edition, McGraw Hill Education, December 2014

If you're thinking about becoming a professional pen-tester, this is the book for you. It covers most of the topics that any ethical hacker should master: basic programming skills, static and dynamic code analysis, fuzzing, an introductory chapter on writing Linux shell-code, advanced exploits and web hacking. It also features some chapters on Windows exploitation, 'Man In The Browser' attacks mainly focussed on IE with BeeF, and some interesting chapters on reversing Android malware using decompilers.



2 Advanced Penetration Testing, Wiley, April 2017

Once you are comfortable with basic penetration testing skills, why not delve deeper into Advanced Persistent Threats (APTs)? This book is an amazing guide to advanced pen-testing techniques, discussing highly secured scenarios and different ways (mostly based on social engineering tactics) to entice users. What's really great about this book is that it shows you more than just creating exploits which are Metasploit-like. Its step-by-step approach for building a reliable, encrypted Command and Control (C2) infrastructure from the bottom up is just awesome. In short, this is a must-have!

Alexander Smith
is a computational physicist. Alex teaches Arduino to grad students and discourages people from doing lab work manually

Resources

- Arduino Mega
- Adafruit Electret Microphone
- Resistors 25x 10k
- SD card adaptor
- A pair of headphones (without buttons)
- Soldering iron

Tutorial files available:
filesilo.co.uk

PART ONE

Build a sound recorder and player using Arduino

In the first part of this tutorial series, we'll discover how to record audio, save it to disk and play it back

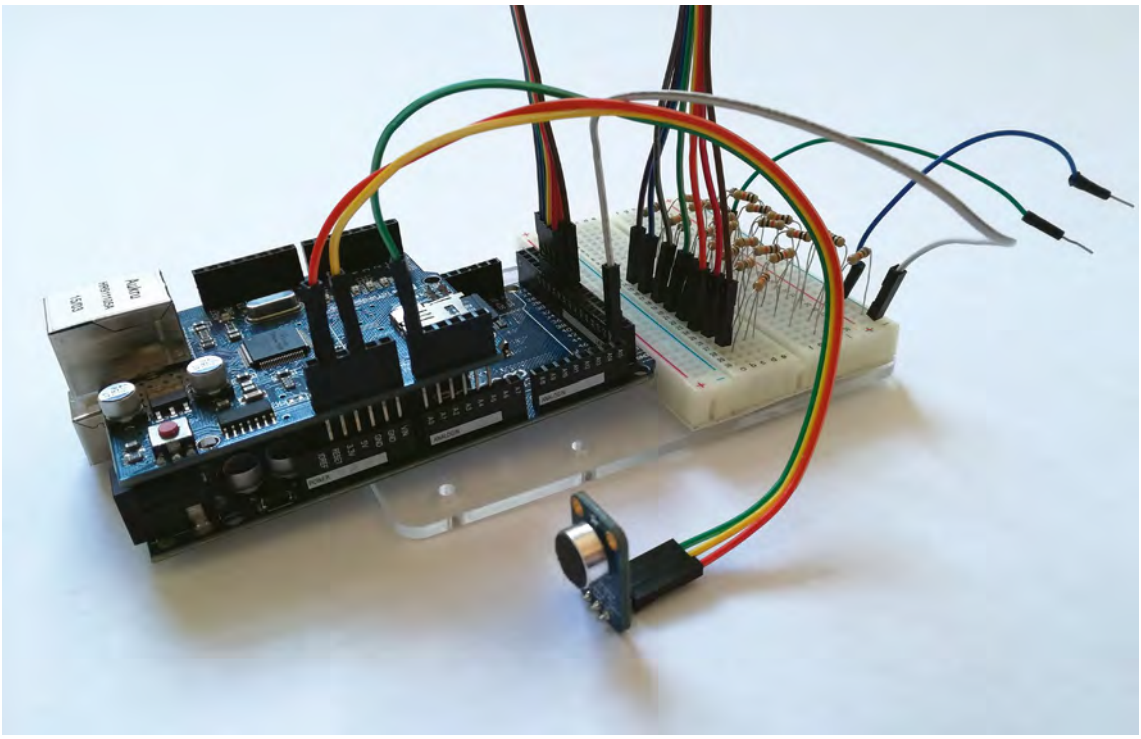
In previous tutorials, we've created niche devices for deployment around the home, but there's no reason that we can't use Arduinos to power real-world products. Microcontrollers have been used in devices such as microwave ovens, remote controllers and computer keyboards – applications where a computer would cost too much money and power. In this two-part tutorial, we're going to take an Arduino, add input and output hardware, create some circuitry and construct a finished product: the rough equivalent of a Dictaphone.

We'll begin by connecting the microphone and try to measure an audio signal. By adding an SD card adaptor, the Arduino will be able to save the sound data to disk. We'll then build a small circuit (using only resistors) to convert 5V digital back to analogue, read that sound file back from the disk and play it aloud on a set of headphones. We'll also discuss ways of improving the sound quality of the input and output audio. In the second part of the tutorial, we'll then add buttons and an LCD display, so the user can cycle through and play stored tracks, begin a new recording and delete old ones.

Connect the microphone

Begin by obtaining an electret microphone and connecting it to your Arduino. In this tutorial, we use the Arduino Mega (for its surplus output pins) and the Adafruit Electret Microphone Breakout – which handles the amplification and noise reduction for us. However, the additional circuitry is relatively simple and you could pick up a microphone for less than £1 online if you wanted to construct the entire circuitry from parts.

The Adafruit microphone doesn't come with ready-to-use input and output pins – so you'll need to undertake a small amount of soldering to create reliable connections. Trying to cleverly bend wires to create a firm connection won't work for such sensitive analogue signals, trust



us. Instead, buy a few header pins, solder them to the microphone board, and then add jumper wires to the pins. These should be connected to the Arduino's 5V, ground, and an analogue input pin.

Record a signal

Create a new sketch and set up your analogue input pin using `pinMode(A0, INPUT)`, changing '0' to correspond to the pin you've used. Also declare integers called `recording` and `playback` and set `recording` to `1` and `playback` to `0`.

In `loop` you should then be able to create a `while` loop for when the device is in 'record mode'. Inside this loop use `analogRead(A0)` to get a measure of the voltage at that instant. This voltage signal will be centred at half the input voltage and fluctuate in time – the audio signal. The Arduino will interpret its value as an integer between 0 and 1023 (a 10-bit value), so you should get a number near 500 most of the time which looks like it's oscillating. If it hardly oscillates, you might need to adjust the gain or add an amplifier. You should also want to triple-check the microphone is connected properly.

If you've not added much code to your sketch, you'll probably be taking measurements with a roughly constant time interval – the sample rate – which during testing was 120 microseconds. This is roughly the time it takes for the analogue-to-digital converter to generate a good average measurement of the voltage. This is the limiting factor of the Arduino sample rate; however, you should still be able to manage 8kHz – the sample rate used in telephone communication to this day.

If the Arduino was only saving a sound file to disk, you would be pretty pleased at how good the device was. However, to turn this into a finished product you're going to need to add additional functionality.

Compress to 8-bit audio

Compressing the audio (and therefore reducing the sound quality) might seem like a silly thing to do. However, you're going to eventually want to play back recordings through a speaker, which can be done very easily on an Arduino using eight digital output pins – or eight bits. Also, conveniently, a byte is eight bits long, which makes writing 8-bit or less quality audio to the SD card very easy. Let's not forget, telecommunications are performed using 8-bit audio at 8kHz – so even by throwing some information away, you should be able to match the audio quality of a phone call.

To turn the 10-bit input into 8-bit, you just need to divide all analogue readings by four. Audio values will now be between 0 and 255 and the sample frequency reduced slightly, accounting for the time the Arduino spends performing the division. While it doesn't matter now, these small pauses taken to perform tasks eventually hamper the maximum sample rate of the device.

Save to SD card

Now that you've got 8-bit audio, you're going to need to store the data somewhere so it can be played back later. The Arduino's internal memory is much too small to store

Above The Adafruit microphone comes with adjustable gain – useful if your sound recordings seem too quiet. It's somewhat sensitive and fragile, though, so be gentle

an entire audio track. Instead, use an SD card and shield [see Tutorials, p44, **LU&D** 184] and write to disk. When you need to play back recordings, the Arduino can simply read the file. If using a shield, you should be able to just insert the legs into the Arduino pins (making sure the six ICSP pins in the middle of the board and shield line up). The procedure is different for the SD adaptor, but

> "Eventually, small pauses taken to perform tasks will limit the maximum sample rate of the device"

there are plenty of diagrams online. Make sure you use a higher-class SD card with decent read/write speed.

In `setup`, initialise the SD card hardware. It is important the program does not proceed if the SD card is not initialised and it should keep attempting to initialise until it does so. This can be done with a while loop:

```
while (!SD.begin() ) {
   Serial.println("SD failed init");
}
```

Declare a file variable globally and assign it an arbitrary filename. Right now, you just want to be able to store a sound file and shouldn't yet mind if it overwrites or appends to an old file. In `loop`, just after reading the analogue signal and converting it to 8-bit, cast the measured signal to a byte and write it to SD.

```
audioByte = (byte) audioLevel;
audioFile.write(audioByte);
```

Writing to the SD card byte-by-byte (literally) is always going to be a very slow process. Having a higher-class

Improved recording

The Arduino power supply on 5V is notoriously noisy when powered over USB, which can greatly affect your microphone performance. Switch to 3.3V power and you should immediately notice better audio quality on recordings. Alternatively, you could try and filter out the noise with a large capacitor. You could also apply a filter to the microphone output.

Higher-quality sound

The sound quality achieved here is reasonable, but can be improved greatly. By combining multiple ports, you can achieve 10-bit sound output. If the resistors in the resistor ladder were changed (they don't have to be R-2R), then a more accurate signal representation could be achieved. A combination of an amplifier and filter at the output will also improve quality noticeably.

SD card should make things better. However, it doesn't help that there are write-size limitations with an SD card so, to write a single byte, the Arduino has to read extra information from the card before writing it back – with the new value appended. What makes matters even worse is that the default setting in the SD library is to physically write to SD after every write statement is called, without making use of a buffer. If this read/write takes too long, it's going to start under-sampling the audio recording. You can get around this by using the SDFat library or by opening the file with some optional arguments, like this:

```
audioFile = SD.open(filename, O_WRITE | O_CREAT);
```

This means you can use the `SD.write(string)` function without hampering your sample rate – just make sure to use the `SD.flush()` command regularly to write a reasonably sized buffer to the card, say after 1,000 bytes.

Within the 'recording' `while` loop, make sure to stop recording after a certain amount of time and set the device to switch to playback mode. You can use the `millis()` or `micros()` command to get the time in milliseconds or microseconds respectively, or you could just count the number of times you flush the buffer.

> "You should be able to recreate telephone-quality sound from an afternoon's work with an Arduino"

In the future, you'll add a button to start and stop recordings, but for now you just need to make a short recording so that you can check the microphone and audio output is working.

Create the digital-to-analogue converter

Now you've got an audio track stored as a series of bytes on an SD card, it's time to try and get the speaker side of the device working. You'll then have the core functionality of the 'Dictaphone' up and running – the rest will just be to make the device more user-friendly.

To output sound, you'll read the recorded voltage levels and you'll want to pass that voltage to an external speaker. There is one small problem: you want to do this using digital pins – which are all at 5 volts (or 0 volts). How can we represent values between 0 and 255 using only a series of binary pins? We can use a byte, of course! You can set the digital pins to match each byte read from the SD card. By using an R-2R resistor ladder (a set of adjoining voltage dividers), you can make each digital pin – each bit – represent twice as much voltage as its neighbour, for all rungs on the ladder, all the way up to one byte.

Begin with a clear breadboard and place seven resistors (with a single resistance) in series down the board, so that their legs share a row with their neighbours. These are the 'R' resistors. We used 10k resistors for this. An extra resistor should be placed last in the series with twice the resistance value (20k, or '2R') and should be connected to ground. The front leg of the first resistor will connect to the speaker. There are many great diagrams online if you get stuck – we've also included a photo to help (see facing page).

There should be seven rows on the breadboard at which resistors meet. At these points, as well as at the speaker end, connect a new 2R resistor (or two in series with the same, R, resistance) to an unused row on the breadboard – ideally across the divide in the middle. These eight resistors you've just placed now need to be connected to the Arduino. However, they need to be connected to a specific set of pins and in a specific order.

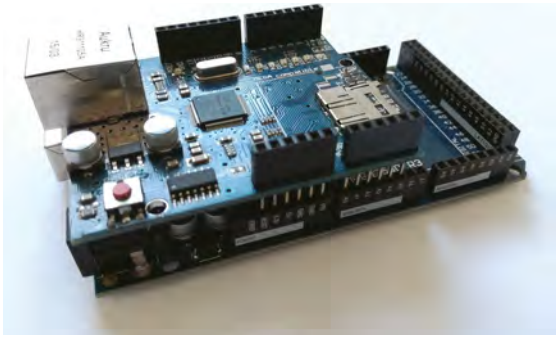
Program your speaker

When the device is in playback mode it should open the designated file and, while the file hasn't run out of content, the next byte should be read. It should then correspondingly set the eight digital pins connected to the resistor ladder to `HIGH` or `LOW`. As simple as it seems, doing this task in sequence is physically slow. A slow process is bad news; not only might it limit the maximum sample rate that can be played, but, if too slow, it could greatly distort the sound output.

Luckily, there is built-in functionality to make a series of eight pins change their state all at once: port manipulation. By setting a port equal to a byte in your sketch, the Arduino chip will find the corresponding pins and set them to their appropriate bit value. This is why it was important to keep the sound quality to 8-bits.

On the Uno and Leonardo, one should take care when setting ports. It is possible to turn off the serial connection, as the only 8-bit-long port uses pins 0 to 7 (which includes transmit and receive). However, on the Mega, there are a couple of ports which are completely out of the way. PORTC, for example, controls pins 30 to 37, is truly digital (not pulse-width modulated) and doesn't share pins with the SD shield.

Connect your resistor circuit to pins 30 through to 37, with the highest-value bit (the resistor furthest from ground) connected to pin 30. As you go down your resistor ladder, you should go up in pin number until you reach 37. In your `loop` function, you can then add the playback code, so that it looks something like this:



Right The Arduino Mega should provide plenty of pins for extending the project next issue

```
if (playback) {
audioFile = SD.open(filename, O_READ);
while (playback && audioFile.available()) {
PORTC = audioFile.read();
      }
      audioFile.close();
}
```

Connect your headphones

In principle, the prototype should be just about finished. It won't be perfect, but if you add a wire to the ground connection at the bottom of the resistor ladder and another wire at the top of the ladder, you should be able to connect a pair of headphones to it.

Connect the ground wire to the bottom of the outermost casing, and the 'top' wire to the tip of the headphone jack. Put on/in your earphones and hold the wires in place with your hand. You should start to hear some noise, Alexander Graham Bell-style.

If you're lucky, it should sound a little bit like Neil Armstrong stepping out onto the moon. Because it's 8-bit audio, with low sample rate, there won't be much 'depth' to the sound, but you should be able to clearly make out your own voice. If this is working, then you won't need to adjust the hardware any further.

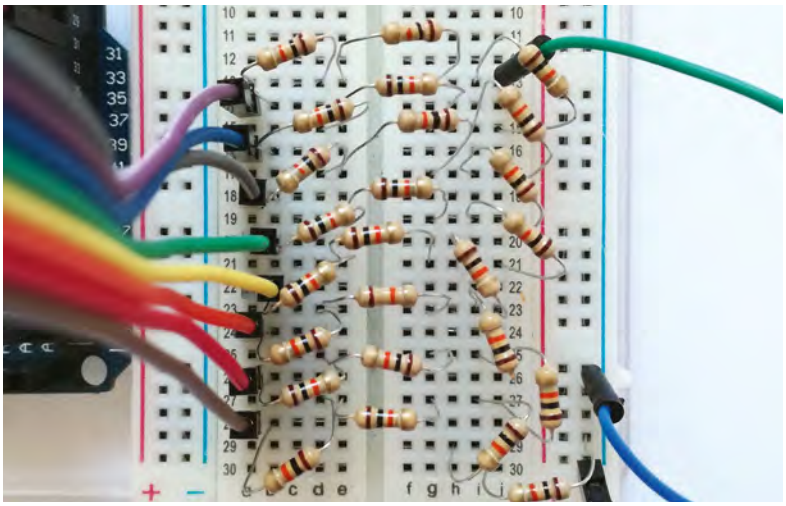
If it isn't, you might need to re-check the connections – yes, all 30 of them if necessary. If the sound is too quiet, you will want to adjust the gain. On the Adafruit board there should be a small knob which can be gently adjusted with a screwdriver. If you've built your own circuit, you might want to consider using transistors or a ready-made integrated circuit. If the recording is too noisy you might need to add a filter.

It's likely that your voice will sound incredibly high-pitched, like someone is pressing fast-forward on an old cassette tape. The reason for this is that it is quicker to read from SD and write to the port than it is to make an analogue voltage reading. So whatever rate it recorded the data at, playback is at a much faster frequency.

In both the recording section and playback section of the loop, it is worth enforcing a sample/playback rate to overcome this. If you've followed the tutorial exactly, you'll probably find that it's impossible to sample higher than 10kHz. There are various ways around this, but for recording the spoken word you can get quite reasonable quality at that rate (remember what we said about telecommunications quality).

To fix a sample rate, calculate the period that needs to elapse between a previous sample and the next one, and don't take a reading or change the speaker state until this period has elapsed. Because you're sampling at kilohertz frequencies, you'll need to use `micros()` rather than `millis()`. This is how you might want to do it for the recording section:

```
while (recording) {
if (micros() - lastSample > samplePeriod) {
              lastSample = micros();
              audioLevel =  analogRead(A0);
      …
```



Above The R-2R resistor ladder is well-documented online and elsewhere. ICs are available cheaply for purchase online if you don't want to construct your own

While your device might now be able to reliably play back a recording of your voice, it's likely that it isn't doing a very good job of it. For a start, there are better ways of guaranteeing a constant sample and playback rate. If you want to spend the extra time, it could be worth adding a timer interrupt which triggers recording or playback events depending on which mode the device is in. This means, regardless of what the Arduino is doing, it will adjust the speaker or measure audio signal on time.

It's also a good idea to switch from 5V to 3.3V power supply if you haven't already. While this will lower the sensitivity of the microphone, it will also lower the noise, as the 5V supply from the Arduino is much 'noisier' than the 3.3V supply. It's surprising how marked this effect is.

Increasing quality

You might also have noticed some repetitive clicking while listening through your headphones. This high-frequency noise can be reduced further by adding a low-pass filter – a capacitor between the 'top' resistor and ground – also known as a reconstruction filter. The ideal capacitor will depend on the resistors you have used, as well as the components used in your headphones. Because of this, it might be easier to just try a few low-valued ceramic capacitors than trying to calculate it.

For a more accurate representation of your voice you can always increase the sample rate by making the analogue-to-digital converter capture the audio input faster. To do this you could adjust the ADC clock frequency or change the number of clock cycles needed to make a conversion. This has been noted online and is discussed on the datasheet for the microcontroller chip – which also covers other useful features such as noise cancellation and gain.

Regardless, by this point you should be able to recreate telephone-quality sound from an afternoon's work with an Arduino. In the next part of the tutorial, we'll turn this basic device into a more user-friendly system, with buttons for operating the device and an LCD display. ■

John Gowers
is a university tutor in Programming and Computer Science, with a strong focus on Java. He likes to install Linux on every device he can get his hands on.

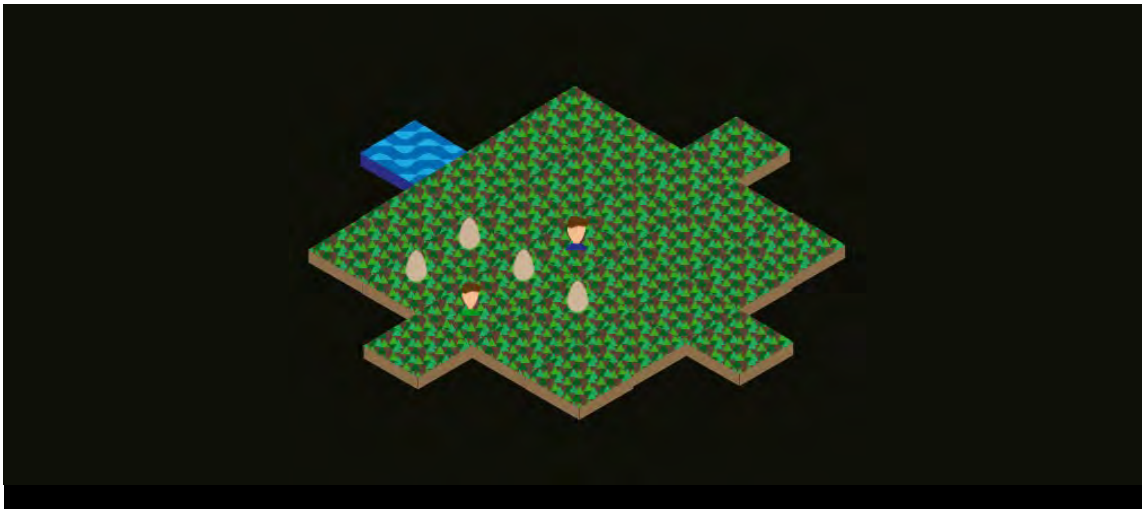
Resources

- **OpenJDK 1.8**
  See your package manager or download from **openjdk.java.net**
- **JavaFX 8**
  See your package manager or download from **openjdk.java.net/projects/openjfx**
- **Eclipse IDE**
  See your package manager or download from **eclipse.org**
- **JGraphT library (optional)**
  Installation instructions in article

PART SIX

Advanced concurrency and artificial intelligence

Writing an automated bot to play our game will help us to learn about some of the new concurrency features in Java 8



Welcome to the last article in this series! If you have been following from the start, congratulations for making it this far. If you are joining us for this last article, don't worry: there's still plenty more to learn.

This series has been about creating a game in Java, but the focus has always been to learn about different Java features. In this article, we will be learning about some of the new concurrency features in Java 8 by writing an automatic bot that can play the game we've been working on. We've learned some simple concurrency using threads, but, as we shall see, threading alone is not quite enough to solve all our concurrency problems. We will introduce the `CompletableFuture<E>` class from the Java 8 concurrency library and show how we can use it to build a service layer in between our bot's strategy and the underlying server architecture.

An important note for those of you who joined the series last issue (Part 5 of the series): the issue for Part 4 included a section about fixing concurrency bugs in the server, but the code supplied for Part V did not include these fixes. If your program is based on the code supplied with Part V, please replace your versions of the `Player`, `StandardGameModel` and `StreamInputController` classes with the versions supplied on the coverdisc.

Our project this month will be a bit different: this time, rather than extending the game we are going to make a malicious automated bot that travels around the map collecting eggs. The actual algorithm that the bot uses is largely up to you: we will describe how to create a very simple (and not very effective) version in this article, but it's up to you to write a more effective one. Feel free to send your completed projects to **linuxuser@futurenet.com** if you want me to have a look at them.

Ideally, what we'd like to be able to do eventually is to write code that might look a bit like this:

```
move('N');
MapTile[][] mapGrid = search();
List<Point2D> eggPositions =
lookForEggs(mapGrid);
for (Point2D eggPosition : eggPositions) {
   pickUpEgg(eggPosition);
}
```

where we have methods `move`, `search` and so on that perform the communication with the server. However, there is a problem with this approach: all the client is able to do is to send commands to the server and – separately – read messages in from the server, asynchronously. No doubt you can think of ways to solve this problem using only threads: for example, one solution might look a bit like the code in **Figure 1** where a separate thread is

responsible for reading messages from the server and setting the hasServerResponded boolean variable when a message has come in. There are several downsides to this approach: it is awkward to write, it does not lend itself well to writing maintainable code and it requires us either to choose a time (500ms in this example) to sleep for while waiting for the server to respond, or to use a 'busy wait' (removing the Thread.sleep() command) that could starve other threads of the resources they need. The most important reason for not using this particular construct is that it is unnecessary: Java already provides much better abstractions for this behaviour that we can use without having to implement them ourselves.

> "The most important reason for avoiding 'busy waits' is that they are unnecessary: Java provides many better abstractions for this natively"

Although we will not be using it in this article, it is worth mentioning the classical Java way around this program, which is by using monitors. Every object in Java is an instance of the Object superclass, which provides the methods wait() and notify(). If I have some object monitor in my code then I can call

```
monitor.wait();
```

at some point in the code, which will cause the currently executing thread to pause. If, in another thread, I then call

```
object.monitor();
```

then the first thread will resume execution.

Monitors are an acceptable way to deal with waiting for input, but in this article we will be looking at a different abstraction, that of a CompletableFuture. If I have a class Result then the class CompletableFuture<Result> is an abstraction representing an object of type Result that will be available at some time in the future. The major advantage that this has over wait()/notify() is that it is a non-blocking call. Compare the two implementations of the search() method shown in **Figure 2**. If I call the first one from a thread, then that thread is paused until I get a response from the server. If I call the second one then the method returns immediately, but it returns a CompletableFuture<MapTile[][]>, rather than the map grid itself.

This is a much more flexible approach: we do not need to block our thread immediately, and can get on with further processing while we are waiting for the server to respond. When we *do* need the server response, we can call the get() method from the CompletableFuture class, which will return the value if it is available, or wait for it if it is not:

Figure 1

```
public MapTile[][] search() throws InterruptedException
{
   sendCommand("search");
   while (hasServerResponded == false) {
      Thread.sleep(500);
   }
   return getServerSearchResponse();
}
```

Above Without using new techniques, we can use 'busy waits' for the behaviour we want, but this is inelegant and prone to problems

```
CompletableFuture<MapTile[][]> mapGridFuture =
search();

// When we need it...
MapTile[][] mapGrid = mapGridFuture.get();
```

Meanwhile, somewhere else in the code, we will deal with server input when it comes in, using it to call the complete() method of the CompletableFuture class, which will mark the future as having completed with the given value:

```
public void dealWithServerMapTiles(MapTile[]
[] grid)
{
   currentSearchRequest.complete(grid);
}
```

If a thread is waiting for the get() method of that particular CompletableFuture object to return, then calling the complete() method will cause get() to return with the value. If we complete() the CompletableFuture with a value, then any subsequent requests to get() the value will return it immediately.

The CompletableFuture class has over 50 public methods in it, and so far we have only scratched the surface of what it is capable of. We will see some more of what it can do later on.

Getting started

As in previous issues, we've provided a complete set of code from the previous issue for you to get started with. To use it, find the file eggs.tar on the coverdisc and copy it to your system. Fire up Eclipse and navigate to File > Import... in the menu. Select 'Existing Projects into Workspace', then click Browse... and navigate to eggs.tar. Press Finish to finish importing the project.

If you are working from your own code, we have provided an extra package that you might find useful for finding paths between accessible map tiles. In order to use it, download eggs.tar to your system as before. Then, in Eclipse, create a new package in your project with the following name:

```
luad.eggs.network.botClient.pathFinder
```

Figure 2

```
public MapTile[][] search()
{
   sendCommand("search");
   this.wait();
   return getServerSearchResponse();
}
```

```
public CompletableFuture<MapTile[][]> search()
{
   this.currentSearchRequest =
      new CompletableFuture<MapTile[][]>();
   return this.currentSearchRequest;
}
```

Above Completable futures give us a more elegant way to deal with results that are not immediately available, but will be some time in the future

Next, select File > Import... and select Archive File. You should be presented with the dialogue in **Figure 3**. Click Browse... at the top and navigate to `eggs.tar`. A directory tree will appear at the left hand side; untick the box next to the top-level directory and then navigate to:

```
src/main/java/luad/eggs/network/botClient/pathFinder
```

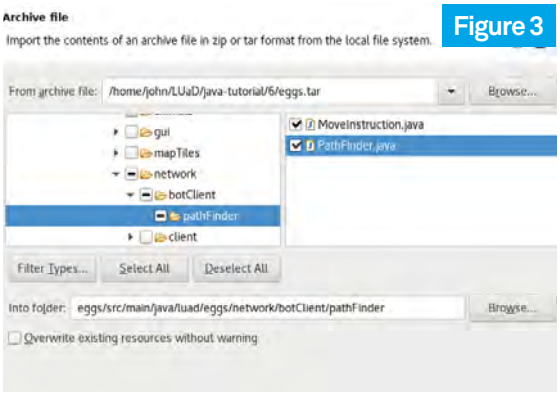
Tick the boxes next to the files `PathFinder.java` and `MoveInstruction.java` at the right and then, under 'Into folder...' further down, click Browse... which should bring up the dialogue in **Figure 4**. After selecting the package that you just created, press OK and then Finish to import the classes.

This should import the new classes into your program. However, the new classes rely on a Maven dependency which you will need to install before you can use them. Follow the procedure that we used to install Spring in the last issue: double-click `pom.xml` in the Project explorer, go to the Dependencies tab and click Add, bringing up the dialogue shown in **Figure 5**. The dependency we want to add is a graph algorithm library called JGraphT. Under Group Id, type `org.jgrapht`, under Artifact Id type `jgrapht-core` and under Version type `1.1.0`. Click OK and then run the `eggs install` run configuration that we created in the last issue. If you have lost the run configuration somehow then you can recreate it by selecting Run > Run configurations... and creating a new Maven build for this project with the goal `install`.

Lastly, as mentioned those of you who joined the project last month need to download some modified versions of some of the basic game classes. Following the instructions above for importing the `pathFinder` package above, extract the files `StandardGameModel.java`, `Player.java` and `StreamInputController.java` into the `luad.eggs` package, replacing the existing versions of these files. You might want to save backups of the originals if you have substantially changed them.

Create a new package called `luad.eggs.network.botClient`. The first class we are going to create will be the service layer class, which will perform two roles.

Figure 3



Top right The 'Import Archive File' dialogue in Eclipse allows us to import individual packages into an existing project

Firstly, it will replace the GUI from the original client: rather than displaying information from the server graphically, it will insert it directly into the bot's 'brain'. Secondly, it will be responsible for sending messages back to the server via the observer pattern. Create a class in the new package called `BotServiceLayer`, making sure that it `extends` the `Observable` class and `implements` the `OutputViewer` interface. This class will contain methods that the bot can use to get information from the server. We will walk you through the process of creating one of these and then let you create the rest along the same pattern.

Inside the `BotServiceLayer` class, create a method called `requestMapTiles()` that takes no parameters and returns an object of type `CompletableFuture<MapTile[][]>`. This method will work as follows: it will instantiate a new blank `CompletableFuture<MapTile[][]>` object and immediately return it. Meanwhile, the class will keep a reference to the `CompletableFuture` we have created; when the server sends across some map tiles, it will use them to complete the future. Since we need to keep a reference to the

A brief history of Java concurrency

Java was one of the first languages to support concurrent programming natively. The Java Virtual Machine provides support for threads, which are the basic unit of concurrency in Java. The `wait`/`notify` mechanism, for pausing threads while they waited for some action to complete in another thread, has been part of Java from the start. But it's a fairly limited mechanism, and it is not easy to avoid bugs – particularly if we accidentally leave out a `notify()` call somewhere. The release of Java 5 introduced the `java.util.concurrent` package, which brought many new, more powerful, concurrency features into Java. This package introduced `ExecutorServices`, service objects which we can submit tasks to have them run asynchronously in a thread pool; data structures such as `BlockingQueue` and `ConcurrentHashMap` that were built to work with concurrency; explicit synchonization locks; and the `Future<E>` interface, which provides a more limited version of the functionality present in the `CompletableFuture<E>` class we are meeting in this issue.

Java 7 introduced the `ForkJoinPool` class, an implementation of the `ExecutorService` interface that allowed computations to be 'forked' to two separate threads and then 'joined' back when they were complete. The `CompletableFuture` class itself was the main new addition to the concurrency library in the Java 8 release. If you complete this tutorial, you'll be at the cutting edge!

future, create a field in the `BotServiceLayer` class of type `CompletableFuture<MapTile[][]>` called `currentMapGridRequest`. Inside the `requestMapTiles()` method, send the command set this field by creating a new `CompletableFuture<MapTile[][]>` object, and then return the field:

```
currentMapGridRequest =
   new CompletableFuture<>();
return currentMapGridRequest;
```

This method should also send the `"search"` command to the server. In order to send commands, use the `setChanged()` and `notifyObservers()` methods from the `Observer` superclass; we will later be adding, as an observer to this class, a class that will read these values and pass them to the server.

```
setChanged();
notifyObservers("search");
```

Now we have to deal with completing the future when we hear from the server. If you haven't already, create the `displayMapTiles()` method from the `OutputViewer` interface. This is the method that is called when the server sends over map tiles. Inside this method, complete the future using the map tiles sent from the server:

```
@Override
public synchronized void
displayMapTiles(MapTile[][] grid)
{
   currentMapGridRequest.complete(grid);
}
```

One slight complication is that the server often sends information that was not in response to a command from the client. For example, the server will send out new map tiles if a nearby player made a move. `CompletableFuture` provides a very good way round this problem: once a `CompletableFuture` has been completed, any subsequent calls to `complete()` will have no effect. What this means is that the first set of map tiles we receive after sending out the command will be used to `complete()` the future and any subsequent map tiles will be ignored.

The only possible problem is that the server might send map tiles before we've had a chance to ask for any. In that case, if the `currentMapGridRequest` field has not been initialized, we will run into an error. In order to ensure that this does not happen, initialise the field when you create it:

```
private CompletableFuture<MapGrid[][]>
currentMapGridRequest =
      new CompletableFuture<>();
```

Now do the same thing for `"inventory"` and `"pick up"` requests. Create methods `requestInventory()` and `requestPickUp()` that return `CompletableFuture`s – a `CompletableFuture<Player>` in the case of `requestInventory()` and, in the case of `requestPickUp()`, a `CompletableFuture<Boolean>`, where the Boolean return value should be `true` if we picked up an egg successfully or `false` if we failed to pick up eggs. Implement these methods exactly as we implemented `requestMapTiles()`, creating new `CompletableFuture` fields `currentInventoryRequest` and `currentPickUpRequest` to hold the return values. You should call the `"inventory"` and `"pick up"` commands in these methods, in the same way as we called `"search"` before.

In order to complete the futures, we use the `displayInventory()` and `displayMessage()` methods from the `OutputViewer` interface. `displayInventory()` can use its parameter to complete the future straight away, as with the `displayMapTiles()` method. In order to complete the `currentPickUpRequest` future, we need to condition on the message received as a parameter to the `displayMessage()` method. When the client sends the `"pick up"` command, the server will respond with a message depending on whether the pick up was successful or not. Depending on what message is passed in as a parameter to the `displayMessage` method, complete the future `currentPickUpRequest` with either `true` or `false`. If the message from the server does not relate to the success or failure of trying to pick up an egg, then do nothing.

You should also create a method `move()` in the `BotServiceLayer` class that takes in a character (`'N'`, `'S'`, `'E'` or `'W'`) and sends the command `"move N"` (or `"move S"` and so on) to the server. If you like, you can make this method return a `CompletableFuture` depending on whether the move was successful or not, but it is simpler to make it an ordinary `void` method. As in the other methods, use `setChanged()` and `notifyObservers()` to send the command.

Figure 4

Left If you want to use our `pathFinder` classes, don't forget to create the package first before copying the classes into it

Using Spring for this assignment

If you completed the project for the last issue, then you might be keen to use the Spring framework again this time round. This is a great idea – there are plenty of places in this program where you can use dependency injection. The only problem is that Spring doesn't behave very well if you have multiple `@SpringApplication`-annotated classes in the same project. A possible solution is to read up about XML configuration – that way you can configure both the server and the bot client in separate XML files.

Spring asynchronous methods

Spring is far more than a dependency injection framework. One useful tool is the `@Async` annotation, which specifies that a method should be performed asynchronously: in its own thread. One advantage of this over `new Thread(runnable).start();` is that the method can take parameters. To use `@Async`, we annotate our main class with the `@EnableAsync` method. We also provide a bean of type `TaskExecutor` – this is an interface provided by Spring that represents an object that holds a pool of threads and can use them to run different methods asynchronously.

Creating the bot strategy

Now that we have created the service layer, we are ready to write the bot strategy itself. Create a class called `BotStrategy` with a field `serviceLayer` of type `ServiceLayer`. It should be possible to set this field by passing the value in as a parameter to the constructor of the class, so that we can create an instance of the class as follows:

```
BotServiceLayer serviceLayer = new
BotServiceLayer();
BotStrategy strategy = new
BotStrategy(serviceLayer);
```

The important method in this class will be called `run()`, and it will be a `void` method that runs the strategy. This is the point where you need to be creative and make your own AI for the bot! In this article, we'll describe a simple version in order to illustrate how to use the `CompletableFuture`s that the service layer returns. Our initial version of the `run()` method is shown in **Figure 6**. There are two new methods that we have written in order to make this method work: `findAndPickUpEgg()` and `moveInRandomDirection()`. The second of these is fairly simple to implement: we randomly choose one of the characters `'N'`, `'S'`, `'E'` and `'W'` and pass it to the method `serviceLayer.move()`.

The first method is also fairly simple to implement, if you use the `pathFinder` package that we have provided with this month's code. First, if we have an two-dimensional array `mapGrid` of `MapTile` objects, we can

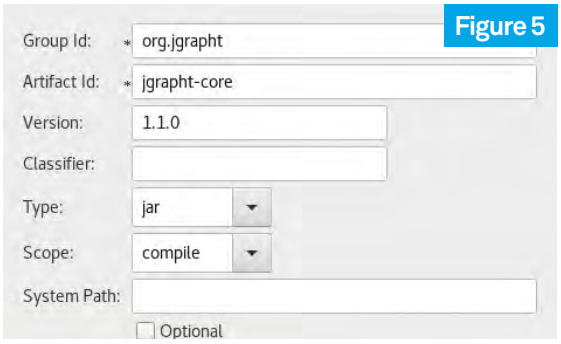


Figure 5

Right The `pathFinder` package relies on the JGraphT library, which we can import through Maven using Eclipse

Below The service layer we have created using `CompletableFuture` allows us to write much more readable code

Figure 6

```
public void run()
{
  while (true) {
    MapTile[][] mapGrid =
      serviceLayer.requestMapTiles().get();
    boolean foundEgg =
      findAndPickUpEgg(mapGrid).get();
    if (!foundEgg) {
      moveInRandomDirection();
    }
  }
}
```

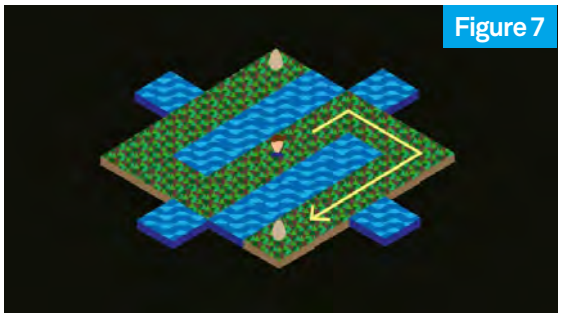


Figure 7

Above The `pathFinder` package uses Dijkstra's algorithm to find the shortest path to an egg that avoids inaccessible tiles

create a new `PathFinder` object using:

```
PathFinder pathFinder = new
PathFinder(mapGrid);
```

The `PathFinder` class then provides two useful methods. If we have two `Point2D` objects (corresponding to array indices in the array `mapGrid`), then we can get the sequence of moves we have to make to get from one to the other by using:

```
Point2D playerPosition = new Point2D(3, 3);
Point2D eggPosition = new Point2D(4, 5);
List<Character> moves =
  pathFinder.findPath(playerPosition,
eggPosition);
```

The method will return a list of characters (for example, `['N', 'N', 'E', 'S', 'S']`); if we call the corresponding `"move"` commands in order, it will cause the player to travel from their current position to the tile corresponding to `eggPosition`. The path will automatically avoid any inaccessible (sea) tiles, as in **Figure 7**. If there is no path that does not go through the sea, then the `findPath()` method returns `null`.

Note that the points `playerPosition` and `eggPosition` are relative to the array `mapGrid`: if the grid is what the server returns in response to the `"search"` command, then the player position will always be the point `(3, 3)`.

Since the map grid might contain multiple eggs, we can also call the `findPath` method by passing in a `Collection` or `List` of `MapTile` objects as the second parameter. For example, if we call

```
List<Point2D> eggPositions = ...
List<Character> pathToNearestEgg =
  pathFinder.findPath(playerPositions,
eggPositions);
```

then the `PathFinder` object will automatically return the path to the nearest egg. If there is no path to any of the target positions, then the `findPath()` method simply returns `null`.

The method `findAndPickUpEgg()` that we used repeatedly calls the `move()` command to move along the path returned by the path finder, and then calls

requestPickUp(), returning the CompletableFuture<> object that that method returns.

Have fun creating your own version of the strategy for the bot. Ours is very simple, but we're sure you can create something better!

There is just one thing that we need to do before we wire everything together: when we created our server, we built in a mechanism that ignored client input if it occurred too frequently. Specifically, the server will ignore any message sent less than 100ms after the previous one. We don't want our bot's messages to be ignored, so make sure to add a delay (using Thread.sleep()) of at least 100ms every time the bot sends a command to the server.

Of course, there's always the possibility that our first message will arrive late and that the server will end up ignoring the second message despite the delay we added. In that case, our CompletableFuture will never complete. In order to safeguard against this, we can use the timeout version of the get() method. For example:

```
MapTile[][] mapGrid = serviceLayer
      .requestMapTiles()
      .get(1000, TimeUnit.MILLISECONDS);
```

This version will wait at most a second for the future to complete. If it does not complete in that time, then get() will throw a TimeoutException, which we can catch in order to try again in case the server swallows our messages.

Running the bot

The last step is to put everything together. We create a class BotClient with a main() method that sets up and runs the program. This main() method is based upon the start() method in the EggsClient class, except that we have replaced the GUI with the bot service layer and game-playing strategy.

Skip the first few lines of the EggsClient.start() method, since all they do is set up the GUI, and move on to setting up the networking. Your code might look a bit like the code in **Figure 8**: it is very similar to the networking code in EggsClient.start(), but we use the BotServiceLayer object serviceLayer as our OutputViewer implementation instead of the GuiOutputViewer object we used before.

To finish our main() method, add the following lines:

```
BotStrategy strategy = new
BotStrategy(serviceLayer);
serviceLayer.addObserver(clientHead);
strategy.run();
```

which will set up both the output back to the server and the bot strategy itself. If you are working from your own code, you might have to change this a bit to make it work.

Testing out your bot is simple: start up the server, then start up a normal eggs client (so you can see what's going on) and lastly start up a bot client. Try and follow the bot around the map with your player to check that it's working correctly.

Since this is the end of the series, we want you to have fun with this project. Try and see what interesting bot behaviours you can come up with. And if you feel like going back and enhancing the GUI or another part of the program, you can always go wild with that too. Get creative – and thank you for taking part!

Non-blocking calls

So far, our calls are 'blocking': when we get a CompletableFuture object, we immediately call get() on it, which causes the thread to suspend until the future completes. The true power of a CompletableFuture comes from its ability to schedule events in a non-blocking manner, which can be very useful for preventing concurrency bugs. Some of the most important non-blocking methods of the CompletableFuture class are shown in **Figure 9**. For example, we could change the code in Figure 6 to the following:

```
serviceLayer.requestMapTiles()
      .thenCompose(mapGrid ->
findAndPickUpEgg(mapGrid))
      .get();
```

In this case, the call to findAndPickUpEgg() would be scheduled immediately, rather than when the first future had completed. In a more complicated system with lots of threads accessing the same data, this could help avoid lots of problems.

Figure 8

```
BotServiceLayer serviceLayer =
    new BotServiceLayer();
Socket socket = new Socket("localhost",
9009);
MessagePrinter clientHead =
      new MessagePrinter(socket.
getOutputStream());
ServerListener serverListener =
      new ServerListener(socket);
ServerOutputTranslator outputTranslator =
      new ServerOutputTranslator(serviceLa
yer);
serverListener.
addObserver(outputTranslator);
new Thread(serverListener).start();
```

Left The bot client follows the same pattern as the Eggs client. The difference is that we replace the GUI with the bot service layer

Below The true power of a CompletableFuture comes from its extensive capabilities for non-blocking scheduling of events

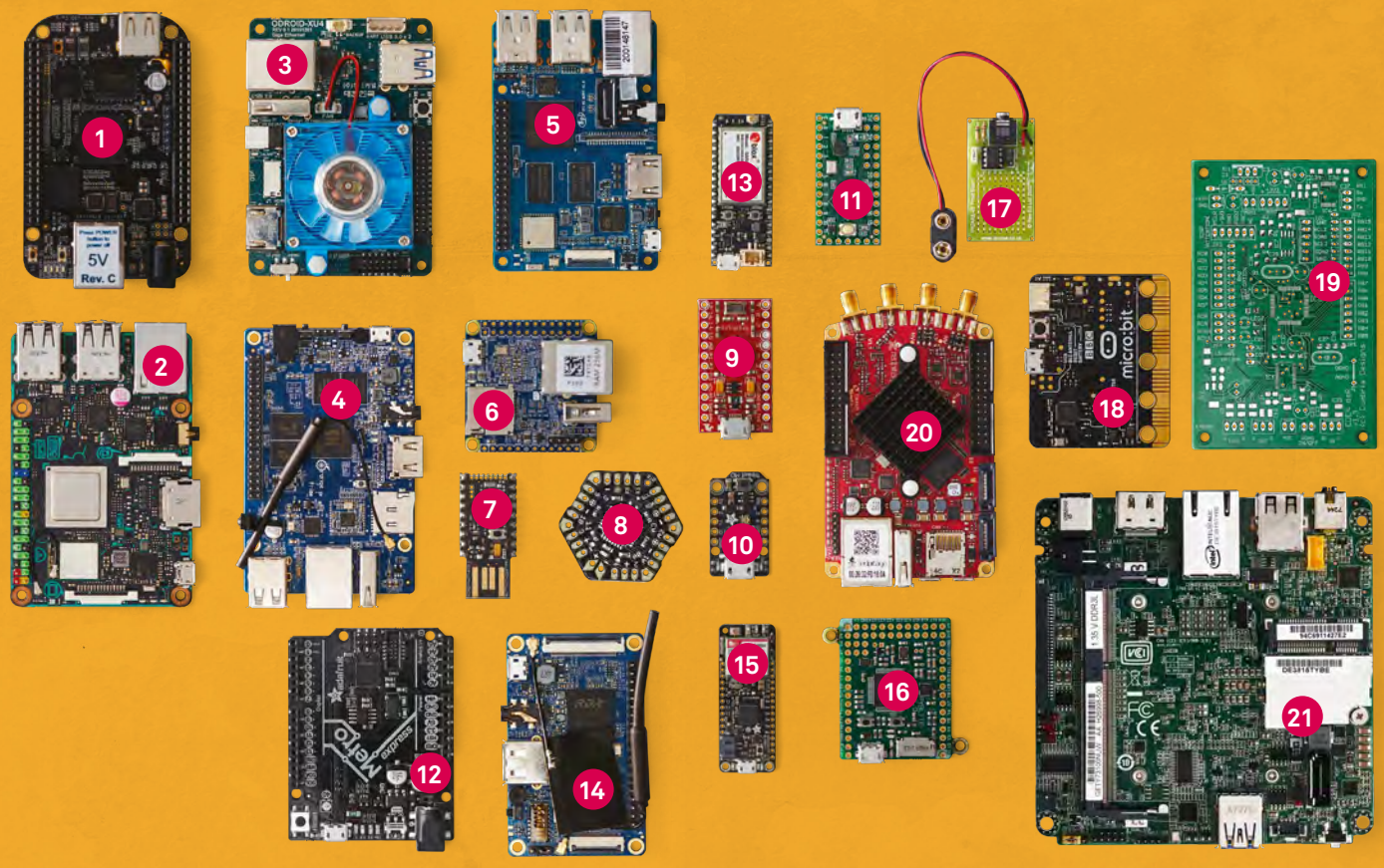
Figure 9

completedFuture()	**Static method that produces a future that is already completed**
thenRun(runnable)	**Runs the given** Runnable **(or lambda) once the future has completed**
thenApply(function)	**Returns a new completable future whose value is obtained by applying the function to the value of the existing future when it completes**
thenCompose(function)	**Like** thenApply**, but now** function **returns a completable future itself**

ROUNDUP

SINGLE BOARD COMPUTERS

Mike Bedford cherry-picks the exciting single board computers on offer, and the many products that exist outside of the world of Raspberry Pi



THE BOARDS

1. Beagle Bore Black Revision C
2. Asus Tinker Board
3. Odroid XU4
4. Orange Pi Prime
5. Banana Pi M2 Berry
6. NanoPi Neo
7. Espruini Pico
8. DFRobot µHex
9. SparkFun Pro Micro
10. Adafruit Trinket M0
11. Teensy LC
12. Adafruit Metro M0 Express
13. Particle Electron 3G
14. Orange Pi 2G-IOT
15. Adafruit Feather FONA
16. MicroPython pyboard lite v1.0 with accelerometer
17. PICAXE-08M2 chip & PICAXE-08 Prototyping Board
18. BBC Micro:bit
19. Cumbria Designs Eden DSP
20. Red Pitaya
21. Intel Thin Canyon NUC Atom E3815 1.46GHz

AT A GLANCE

Types of single board computers

- **Pi-like boards** p60
Boards such as the Raspberry Pi are ideal both for learning to code in a Linux environment, and for interfacing to real-world electronics and equipment.

- **Arduino-like** p62
Arduino-style boards differ from Pi-like versions in that they don't run an operating system. For embedded applications – that is, ones that involve monitoring or controlling external hardware – this offers several benefits.

- **High- and low-powered** p64
Not all SBCs are Pi-like or Arduino-like. High-powered boards use advanced or even esoteric hardware for increased performance, perhaps for specialist applications. Low-powered products either provide a very simple learning environment, or allow a bare-bones solution for compact embedded tasks.

- **Remote IoT boards** p66
Boards for Internet of Things applications could still be Arduino-like, but those with extra connectivity to mobile phone networks are essential for remote monitoring applications.

When it first hit the market in 2012, the Raspberry Pi was considered truly revolutionary, and with some justification. Here was a computer the size of a credit card that did pretty much what you'd expect of a full-sized PC running Linux, but it cost just £22. What's more, thanks to its GPIO header, interfacing it to real-world hardware was much more straightforward than with most PCs.

While the Raspberry Pi caught the public eye almost immediately, another family of single board computers, also aimed at the enthusiast, took rather longer to gain widespread acclaim. Arduino was launched in 2005 and, as we'll see later, was aimed primarily at embedded computing applications. Since the launch of the Raspberry Pi, it too has gone from strength to strength.

Imitation is supposedly the sincerest form of flattery, in which case the Raspberry Pi Foundation and Arduino AG must be over the moon about the number of single board computers (SBCs) now on the market and within reach of the amateur experimenter. While abundant choice is surely good news for the consumer, it also means that choosing an appropriate product can be a daunting experience. This is exacerbated by the fact that not all of the SBCs on the market are exact Raspberry Pi or Arduino clones. Instead, some build on the Pi or Arduino philosophy but add their own twist, perhaps offering additional performance or facilities, while others are intended for very different applications and, accordingly, have very different specifications.

In this feature we'll provide an overview of what's out there, to help you choose something that meets your needs in the most cost-effective way. But we're not necessarily assuming you're fully au fait with the SBC market and are just looking to go beyond the Raspberry Pi or the Arduino. Perhaps you're already a Pi user but haven't delved into Arduino-like boards. We'll explain in broad terms how these two categories of board differ – and the differences truly are fundamental – before discussing the market and putting a few products under the spotlight.

> "While abundant choice is surely good news for the consumer, it also means that choosing an appropriate product can be a daunting experience"

Conversely, you might be an Arduino enthusiast who hasn't really considered the benefits of the Raspberry Pi approach. Again, we'll provide top-level information as well as homing in on specific products. On the other hand, you might have never delved into any form of single board computer, so this overview is exactly what you need. Not all SBCs are either Pi-like or Arduino-like, so we also delve into boards that offer something truly different. Here there are products for those who want to try their hand at programming some quite different hardware and create unique applications.

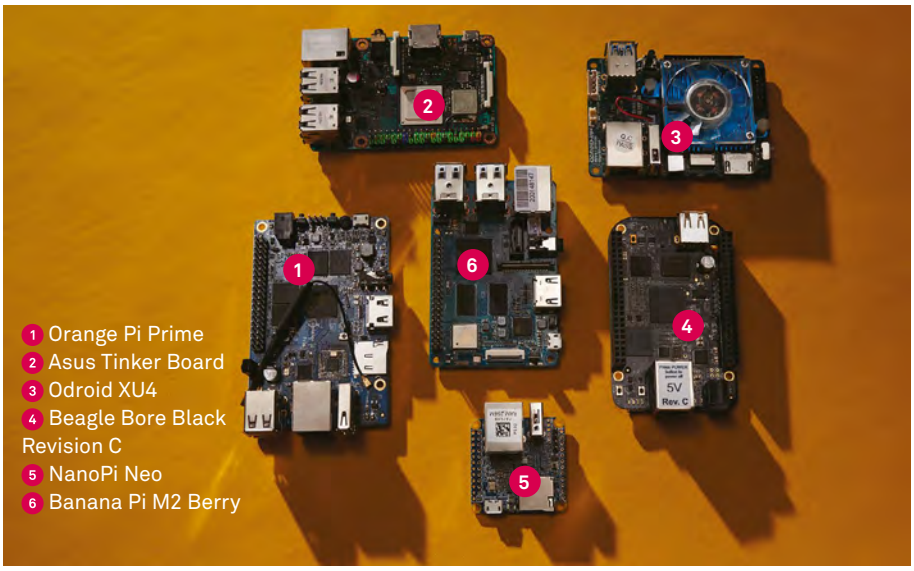
Incidentally, this is not meant to be an exhaustive or definitive list – it's a huge market and all we can hope to do here is scratch the surface. Trying to cover everything the market has to offer would be a daunting task and ultimately quite tedious to read, as many products are basically similar. So while we discuss individual products, it's largely to illustrate the diversity of products available and to give you a flavour of what's on offer - new systems and products are popping up all the time. Use the information provided here for guidance and then when you're ready to dip into the world of SBCs, be sure to undertake a detailed review of the products on your shortlist.

In any case, the indications are that single board computing is here to stay, and whatever your interests, there's likely to be the perfect model for you. So let's discover what these ingenious beasts can do...

BOARDS SUPPLIED BY
PIMORONI NEW IT MOUSER ELECTRONICS

PI-LIKE BOARDS

Whether you're learning to code or intent on interfacing to the real world, a Raspberry Pi or similar will fit the bill



The Raspberry Pi family, and similar boards from other manufacturers, is a jack-of-all-trades solution. It fulfils the two different but related roles of a low-cost Linux computer: a means of learning to program, and a vehicle for learning about embedded computing that interfaces with external hardware. If you're looking for both, or even if you just want to learn to code, this class of SBC is exactly what you need – although if you're only interested in embedded computing, an Arduino-like board might be a better bet.

The key feature of this category is that it runs an operating system and has ports to allow you to connect peripherals such as a keyboard and mouse, a flash-memory card slot, and usually a display monitor. If your sole computing experience involves using a PC, it might seem difficult to believe that some SBCs don't have even these facilities but, as we'll discover when we turn our attention to Arduino-like boards, these are by no means essential for some applications.

The bottom line is that, by adding a mouse, keyboard, memory card, monitor and power supply, Raspberry Pi-like boards allow you to put together a very cost-effective Linux computer. Of course, building a complete PC in this way will cost you more than the headline price of the board – and the incremental cost is greater in percentage terms for bottom-end boards such as the Pi Zero – but, even so, the total price is still very attractive, especially if you can press a TV into service as the display. Note that, while most Pi-like boards can drive a monitor, a few are referred to as 'headless', which means they can't. They still run an operating system but you need to drive them remotely from another PC.

Inputs and outputs

Unlike PCs, an important feature of this class of SBCs is that they have GPIO (General-Purpose Input/Output) pins. The voltage on these pins can either by driven under program control if they're configured as outputs, or the voltage applied to them by external hardware can be read by the software. This is an essential feature for embedded applications – that is, working with external electronic circuitry. Commonly, and certainly when you first start out with embedded programming, you would connect push buttons to GPIO pins configured as inputs, and attach LEDs to pins configured as outputs. A program could then light up an LED when you press a button – although this is just a trivially simple example.

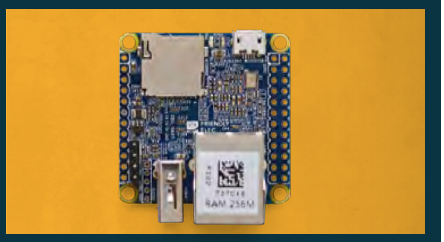
LU&D's top picks

Orange Pi Prime

The Orange Pi Prime (around £33) has surely been designed to compete head-on with the Raspberry Pi 3. The specifications and price are similar (except with more RAM) – although, unlike some Pi look-alikes, the board layout isn't identical. Importantly, though, the GPIO header has the same pin-out for compatibility.

Odroid XU4

Essentially a Pi 3 on steroids, the Odroid XU4 (£73) is the same size as the Pi but has two CPUs at 1.4 and 2GHz with a total of eight cores, and double the memory. Third-party benchmarks tend to favour the XU4 over the Pi 3 by a significant margin. Bear in mind that the GPIO pins aren't Pi-compatible, however.



NanoPi Neo

The NanoPi Neo ($7.99 – around £6) has a lower specification than the Pi Zero despite its higher price. Most significantly, it's headless, so can't drive a monitor. On the plus side, it has a different form factor (being shorter but a bit wider) which could be a plus point for use with some embedded applications.

How the boards compare	Raspberry Pi Zero	Raspberry Pi Zero W	Raspberry Pi Model A+	Raspberry Pi Model B+	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B	NanoPi Neo	Banana Pi M2 Berry	Orange Pi Prime	BeagleBone Black Revision C	Asus Tinker	Odroid XU4
Processor Family	ARM11	ARM11	ARM11	ARM11	ARM Cortex A53	ARM Cortex A53	ARM Cortex A7	ARM Cortex A7	ARM Cortex A53	ARM Cortex A8	ARM Cortex A17	ARM Cortrx A7 & A15
Processor Architecture	32-bit	32-bit	32-bit	32-bit	64-bit	64-bit	32-bit	32-bit	64-bit	32-bit	64-bit	64-bit
Cores	1	1	1	1	4	4	4	4	4	1	4	8
Clock Speed	1GHz	1GHz	700MHz	700MHz	900MHz	1.2GHz	1.2GHz	1.2GHz	1.2GHz	1GHz	1.8GHz	1.4/2GHz
RAM	512MB	512MB	512MB	512MB	1GB	1GB	256MB	1GB	2GB	512MB	2GB	2GB
Onboard Flash	None	None	None	None	None	None	None	None	None	4GB	None	None
USB Ports	1	1	1	4	4	4	2	4	4	2	4	3
Storage Expansion Slot	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD, SATA	MicroSD	MicroSD	MicroSD	MicroSD
Video Output	N	N	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
Camera Port	Y	Y	Y	Y	Y	Y	N	Y	Y	N	Y	N
Ethernet Port	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
Wireless/ Bluetooth	N	Y	N	N	N	Y	N	Y	Y	N	Y	N
GPIO Pins	40	40	40	40	40	40	24	40	40	92	40	42
Dimensions (mm)	65 x 30	65 x 30	65 x 56	85 x 56	85 x 56	85 x 56	40 x 40	92 x 60	98 x 60	86 x 55	85 x 56	83 x 58
Price	£4.80	£9.60	£17.99	£24.99	£28.32	£32.99	$7.99 (£6)	£35.25	£33.79	£45.56	£46.99	£73

Raspberry Pi images: Pimoroni

If you're not familiar with the Raspberry Pi range, the specification of the six current boards is provided in the comparison table. Because we've looked at these products extensively in the past, our main emphasis here is on Pi-like products, so we've included six such boards in the comparison table (see also **LU&D**'s Top Picks box, left).

In terms of specifications, third-party products encompass the complete range, from lower than the Pi Zero to higher than the Pi 3 – although the difference in specifications isn't always reflected in the pricing. Some products can really only be called clones, while others have clearly been designed to offer their own unique take on the Raspberry Pi philosophy. Often those products that aren't just Pi look-alikes have chosen to adopt the same GPIO pin-out, which is essential if you want to be able to attach Pi-compatible HATs (expansion boards) or circuitry you've designed and built for a Pi board. Some, such as the Asus Tinker, have not only maintained compatibility with the Pi but have an ergonomically improved design by colouring the pins according to their function. However, Pi-compatibility isn't universal, so be sure to check up before buying a third-party SBC if this is important.

Though it's tempting, buying more performance than you need isn't generally a good idea, not least as it means shelling out more money. Admittedly, there's a huge difference in the performance of Pi products, with the Zero taking much longer to boot than some of the higher-end products, but for most applications the additional performance offered by some third-party products will go untapped.

For example, some boards such as the Rock64 and the Asus Tinker offer 4K video (3,840x2,160 or 4,096×2,160) compared to the Pi's Full HD (1,920x1,080), and some provide 192kHz/24-bit audio. On the face of it, this would make such boards ideal for home entertainment applications, and this is certainly possible. But we have to question just how many people will use these boards in conjunction with a screen large enough to make the higher resolution noticeable, or an audio system capable of making the most of the higher frequencies.

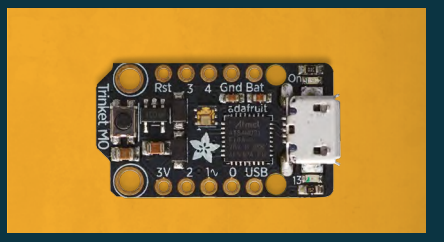
QUICK TIP

Pi, more or less

We've referred to these boards as Pi-like, not Pi-compatible. For embedded applications, using the same GPIO pin-out as a Pi might be necessary. Check before buying any Pi-like board if this is likely to be important.

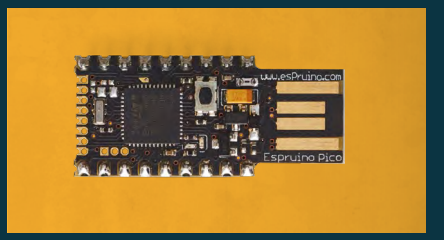
While there are, of course, benefits to be gained by pushing the envelope and embracing some of the newer Pi–like products, we can't help but make a point that is often cited by those who have used SBCs from a number of manufacturers. The point – one that perhaps applies mostly to those intent on interfacing real-world devices to their boards – is that community support is often of more value than the finer points of the hardware specifications. And while things might change in the future, Raspberry Pi products are still the best supported in terms of community advice, inspiration and plain old ingenuity.

LU&D's top picks



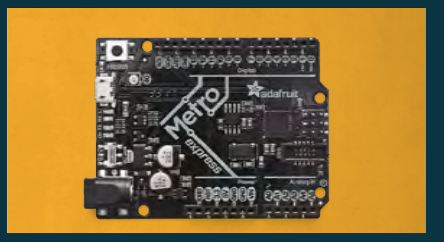
Adafruit Trinket M0

The most remarkable feature of the Adafruit Trinket M0 (£7.50) is that it's so diminutive, despite its high-end CPU. The price you pay for the price you pay, however, is its limited interfacing capability. Analogue I/O, PWM and serial ports all share the same pins as the five digital I/Os.



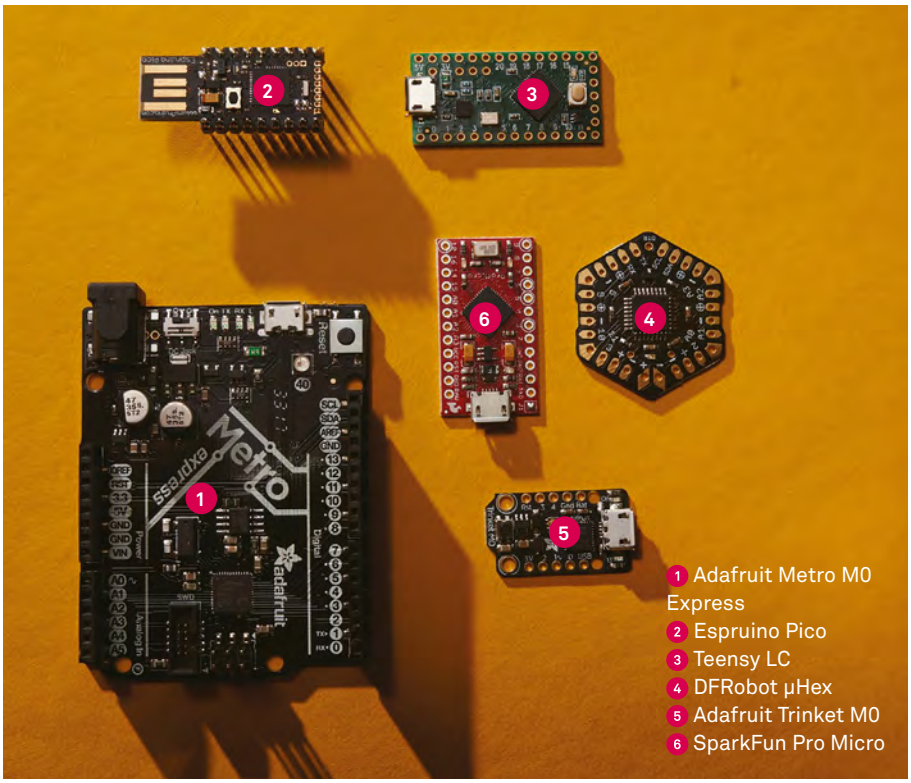
Espruino Pico

Despite the name, the Espruino Pico (£23) is Arduino-like rather than Arduino-compatible. It's a USB stick which you can program using JavaScript from its own IDE simply by plugging it into your PC's USB socket. An adaptor is also available to convert the I/Os for compatibility with Arduino shields.



Adafruit Metro M0 Express

The Metro M0 Express (£23.50) illustrates how you can often get a lot more for your money with third-party derivatives. While the number of I/O pins available isn't particularly spectacular, you get a faster processor and, critically, much more RAM and flash for little more than the cost of an Arduino Uno.



ARDUINO-LIKE

If your application needs to interface with real-world hardware, an Arduino-like board is a strong contender

If you haven't encountered Arduino-like boards before they'll probably appear as something of a mystery, so let's start by setting the scene. First, these boards are intended solely for embedded applications and second, they don't run an operating system, with just the odd exception. So while a Raspberry Pi boots the operating system when you turn it on, a brand new Arduino board will do nothing at all – its behaviour depends entirely on the software in its flash memory. Because there is no operating system, that software has to be developed on a PC and uploaded to the SBC. This works fine for embedded applications; indeed, if this is your sole interest, there are major benefits compared to using a board such as the Raspberry Pi that runs an operating system.

This is because an operating system can represent a huge overhead for simple control applications. If you don't need an operating system, much less powerful hardware can be used. So if you see that an Arduino board has a 16MHz processor, don't jump to the conclusion that it's the poor relation of a Pi board with a 1.2GHz chip. While a slower processor and less memory doesn't always translate to lower-cost hardware, there is certainly the potential for very low-cost platforms to be used for embedded applications. Equally important, if not more so, is the fact that while operating systems can take quite some time to boot before any embedded application can start up, without this overhead the application can start as soon as the board is powered up. If you're questioning the importance of this, consider how popular a car would be if you had to wait a minute for its operating system to boot each time you set off on a journey...

Arduino is open-source hardware, meaning that companies can legally produce clones or derivatives of the products designed by

Arduino. Arduino itself sells quite a few SBCs, and it also licences its designs to several third parties who are then allowed to use the Arduino name and logo. These companies are SmartProjects, Sparkfun and DogHunter. Beyond that, a few manufacturers produce clones but don't use the Arduino name and logo, for obvious reasons. Although such clones aren't certified by Arduino, their reviews are generally favourable and their prices are lower, sometimes much lower.

There are also companies which produce unofficial clones which, nevertheless, feature the Arduino name and logo. This is obviously a breach of copyright and these products should be avoided. Finally, and perhaps most interestingly, there are boards that can be referred to as Arduino derivatives. They're not identical to any particular official Arduino product, but they are still compatible with the Arduino IDE (Integrated Development Environment – the PC software you use to develop software and upload it to the board), and often have pin-compatible GPIO headers. This could be important because, like the Raspberry Pi, Arduino boards are well supported by official and third-party add-ons, which in this case are known as shields.

As with the Pi-like products, we've put together a table comparing six of the more common Arduino boards with a selection

> "These boards are intended solely for embedded applications and therefore don't run an operating system"

of third-party products, all of which can be described as Arduino derivatives, with three particular highlights in **LU&D**'s Top Picks (see left). In general, though, what should you look for in this type of product?

Well, often processor speed isn't too important; an 8MHz 8-bit microcontroller can often perform control applications with ease. The amount of RAM could be significant, however; not because it affects performance as it does with boards that run operating systems, but because it affects the size and complexity of the applications you can write. After all, this class of board might only have a couple of kilobytes of RAM. The size of the flash memory, again usually measured in kilobytes, limits the size of the code.

Possibly the most important feature, though, is the number of external connections a particular board has. These are classified as digital I/Os, some of which can act as PWM (Pulse Width Modulation) channels to provide pseudo-analogue outputs, plus dedicated analogue inputs and outputs. Also bear in mind that while most Arduino-like boards have a USB port to allow their flash memory to be programmed from a PC, a few don't. Leaving out this component obviously cuts down on price, and the lack of it doesn't affect operation when the board is performing its embedded application, but it does mean that you'd need to buy a serial-to-USB converter for programming it.

Lastly, you need to know that Arduino classifies its products as either boards or modules. Boards are physically compatible with shields, while modules – though electrically compatible – will need a converter in order to attach a shield.

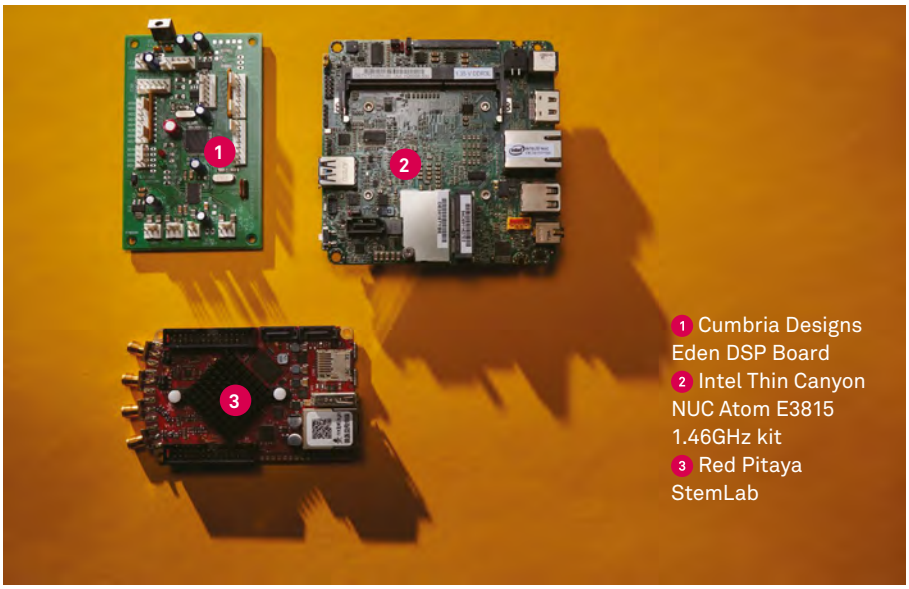
QUICK TIP

Size does matter

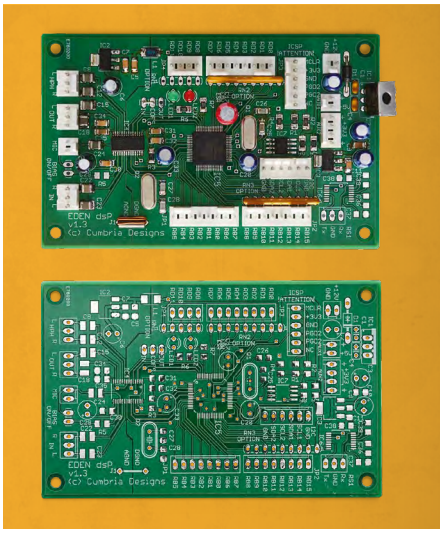
For embedded applications, remember that the physical size of the SBC is often just as important as its processor speed or memory size. After all, the board might need to fit into a particular case once programming is complete.

How the boards compare	Arduino Mini 05	Arduino Micro	Arduino Uno	Arduino Due	Arduino Zero	Arduino Ethernet	Adafruit Trinket M0	DFRobot µHex	Teensy LC	SparkFun Pro Micro	Espruino Pico	Adafruit Metro M0 Express
Microcontroller Core	Microchip picoPower	Microchip picoPower	Microchip picoPower	ARM Cortex M3	ARM Cortex M0	Microchip picoPower	ARM Cortex M0	Microchip picoPower	ARM Cortex M0	Microchip picoPower	ARM Cortex M4	ARM Cortex M0
Processor Architecture	8-bit	8-bit	8-bit	32-bit	32-bit	8-bit	32-bit	8-bit	32-bit	8-bit	32-bit	32-bit
Clock Speed	16MHz	16MHz	16MHz	84MHz	48MHz	16MHz	48MHz	8MHz	48MHz	16MHz	48MHz	48MHz
RAM	2K	2.5K	2K	96K	32K	2K	32k	2K	8K	2.5K	96K	32K
Flash Memory	32K	32K	32K	512K	256K	32K	256K	32K	62K	32K	384k	256K
Digitial I/Os	14	20	16	54	14	14	5	14	27	12	22	14
PWM Channels	6	7	6	12	10	4	2	5	10	5	21	14
Analogue Inputs	8	12	6	12	6	6	3	6	13	9	9	6
Analogue Outputs	0	0	0	2	1	0	1	0	1	0	0	1
USB Port	N	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
Ethernet Port	N	N	N	N	N	Y	N	N	N	N	N	N
Board or Module?	Module	Module	Board	Board	Board	Board	Module	Module	Module	Module	Module	Board
Dimensions (mm)	30 x 18	48 x 18	69 x 53	102 x 53	69 x 53	69 x 53	27 x 11	31 x 28	36 x 18	33 x 18	33 x 15	71 x 53
Price	£ 12.32	£13.77	£15.35	£31.49	£35.99	£39.60	£7.50	£8	£11.50	£22	£23	£23.50

Arduino Images: Arduino.cc



1 Cumbria Designs Eden DSP Board
2 Intel Thin Canyon NUC Atom E3815 1.46GHz kit
3 Red Pitaya StemLab



Above The Eden DSP Board comes as just that – a bare board. When built, it looks more like the top example.

HIGH-POWERED AND LOW-POWERED BOARDS

Some applications require specialised high-performance boards, while others need the ultimate in simplicity or small size. Here we investigate both

Arduino-like boards have sufficient power for many embedded applications, and if you need more computing muscle, Raspberry Pi-type boards often fit the bill. For some applications, though, you need even more power. Basic high-powered boards include products that fall between Raspberry Pi-type boards and conventional PC motherboards. They're much smaller than most motherboards, allowing compact PCs to be put together. A good example is the Intel Thin Canyon NUC Atom E3815 1.46GHz kit. At 102mmx102mm, it's a fair bit larger than a Raspberry Pi 3 but it has a SATA interface for attaching hard drives and a RAM slot so you can configure the memory yourself. It's available from around £112, including CPU.

> "Low-powered does not necessarily mean low-cost; many of these products cost as much, if not more, than fully functioning SBCs"

DSPs and beyond

More specialised products support Digital Signal Processors. DSPs are, like the processors on most SBCs, general-purpose in the sense that they are programmable to do pretty much anything, but have very fast instructions for the types of functions that are used in signal processing. When used with analogue-to-digital converters (ADCs) and digital-to-analogue converters (DACs) they can be used to perform tasks that, until fairly recently, were the domain of analogue electronic circuitry.

For example, while a conventional radio receiver uses quite a lot of analogue electronics, a so-called Software Designed Radio (SDR) can be implemented within a DSP. Generally, DSP boards don't run an operating system so software needs to be developed on a PC and then downloaded to the board.

The ultimate in performance is provided by SBCs that have an on-board FPGA. Otherwise known as a Field Programmable Gate Array, these chips contain logic electronic circuit blocks that can be connected together, under software control, to produce a digital electronic circuit. Given that hardware is generally considered to be faster than software, because it can be much more parallel in operation, this provides a means by which extremely power-hungry applications can be implemented.

As with DSP boards, FPGA boards are equipped with ADCs and DACs and are used in specialised applications for which only an FPGA is good enough. Because an FPGA isn't a general-purpose processor (although it is possible to implement small processors within an FPGA), these boards also have conventional processor cores which carry out standard control functions.

When talking about 'low-powered' boards, we mean one of two things. First, there are boards that are simpler to use than most SBCs, and so are useful for teaching the basics of programming and interfacing to younger students. Second, there are bare-bones solutions that might be little more than a single chip (but with a simple means of programming from a PC) but which are eminently suitable for embedded applications where space is at a premium.

Ironically, low-powered does not necessarily mean low-cost; indeed, many of these products cost as much, if not more, than fully functioning SBCs. If you're looking for a bargain-basement product and the maximum bang for your buck, therefore, you're better off considering some of the cheaper Raspberry Pi-like products.

The simple-to-use category is epitomised by the BBC micro:bit which, in 2016, was given away to every year 7 pupil in England and Wales and used in schools. The main difference between the micro:bit and the Raspberry Pi and its look-alikes is that it doesn't run an operating system, so programs are developed on a tablet, smartphone or PC and uploaded to the board. GPIO is limited but the micro:bit can perform embedded tasks in total isolation thanks to its on-board push-buttons, 5x5 LED array, temperature sensor, compass and accelerometer. The BBC micro:bit costs £13, or £16 for a starter kit which also includes a battery box and the USB-to-micro-USB lead that's needed for programming. A similar solution is provided by the MicroPython pyboard lite v1.0 with accelerometer which has fewer on-board device. At £22.60, it also costs more than the micro:bit.

For bare-bones solutions we're turning our attention to products that use PIC microcontrollers from Microchip Technology. While the family now includes much more powerful products, PIC chips started out by offering circuits that were suitable for very basic control functions, perhaps replacing just a handful of discrete logic chips, but at super-low prices.

These bottom-end microcontrollers are still available and the 12F1840 is typical. It has a 32MHz 8-bit processor, 7KB of flash memory, 256 bytes of RAM and six I/O pins including a four-channel ADC - not much to shout about. However, it's tiny, being housed in an 8-pin package, and it costs £1.26 in one-off quantities, falling to little more than 50p in large volumes.

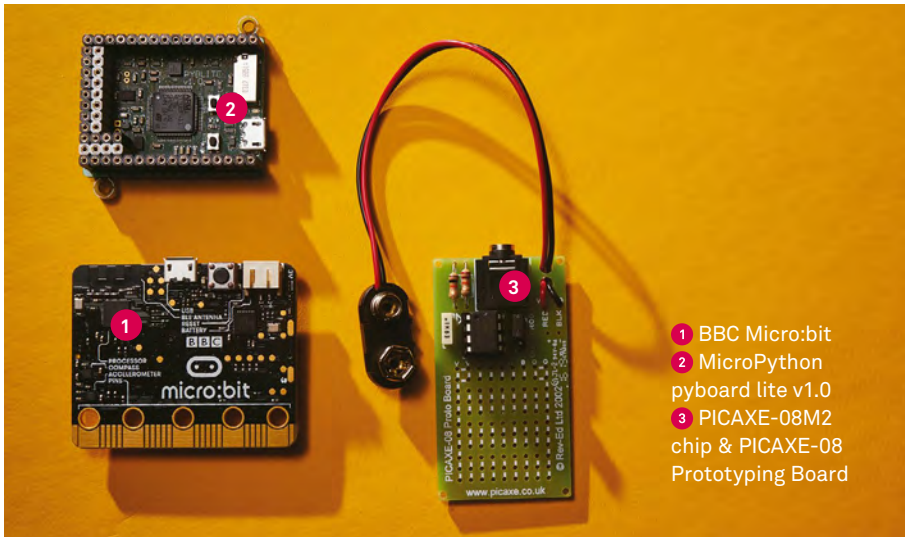
Microchip Technology also offers so-called development boards which allow developers to get some experience of PIC microcontrollers and write code before moving on to design their own hardware using PIC chips. These development boards, such as the £10 Microchip MPLAB Xpress Development Board, can also be used as an SBC by enthusiasts.

The ultimate bottom-end, however, is represented by single-chip solutions – see LU&D's Top Picks box on the right. ▸

QUICK TIP

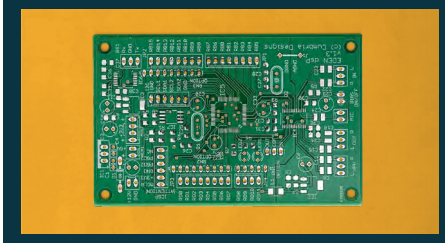
Doing it yourself

If you're working with bare-bones solutions such as the PICAXE, you'll probably want to build it into your own circuits. You'll need tools such as a soldering iron and small pliers.



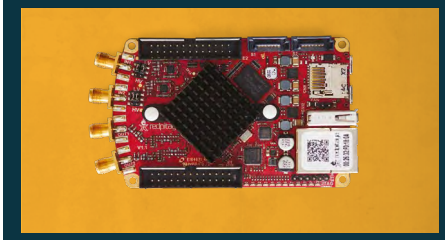
1 BBC Micro:bit
2 MicroPython pyboard lite v1.0
3 PICAXE-08M2 chip & PICAXE-08 Prototyping Board

LU&D's top picks



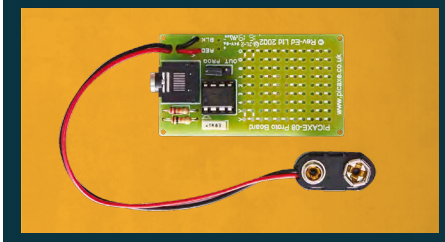
Cumbria Designs Eden DSP Board

This is a bare board (£11) so you have to build it yourself (£25-£30 for components). It supports the dsPIC33EP512MC806 DSP chip for which the development software is freely available. A separate programmer is needed to upload code from your PC. Starter applications are freely available.



StemLab Starter Kit

StemLab (£210) from Red Pitaya contains an FPGA, two ARM processor cores, plus ADCs and DACs. It runs a Linux distribution on-board. It's an open-source project with lots of software freely available – mostly implementing laboratory test equipment such as signal generators and oscilloscopes.



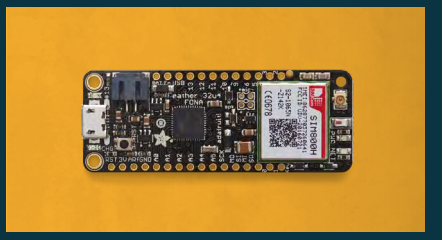
PICAXE-08M2

Not a board but an 8-pin chip, the PICAXE-08M2 (£2) is a PIC microcontroller with custom software for BASIC programming. It's supplied with development software. Although it's intended to be built into your own circuit, the PICAXE-08 Proto Board, as shown in the picture, is also available (£3) to get you up and running quickly.

LU&D's top picks

Orange Pi 2G-IOT

The Orange Pi 2G-IOT (£11) is a Pi-like board, but differs from most in that it doesn't have an HDMI port. Instead, it's capable of driving an LCD panel, such as on a mobile phone. The major difference, though, is its cellular interface which supports 2G and a SIM card. Wi-Fi and Bluetooth are also supported.



Adafruit Feather FONA

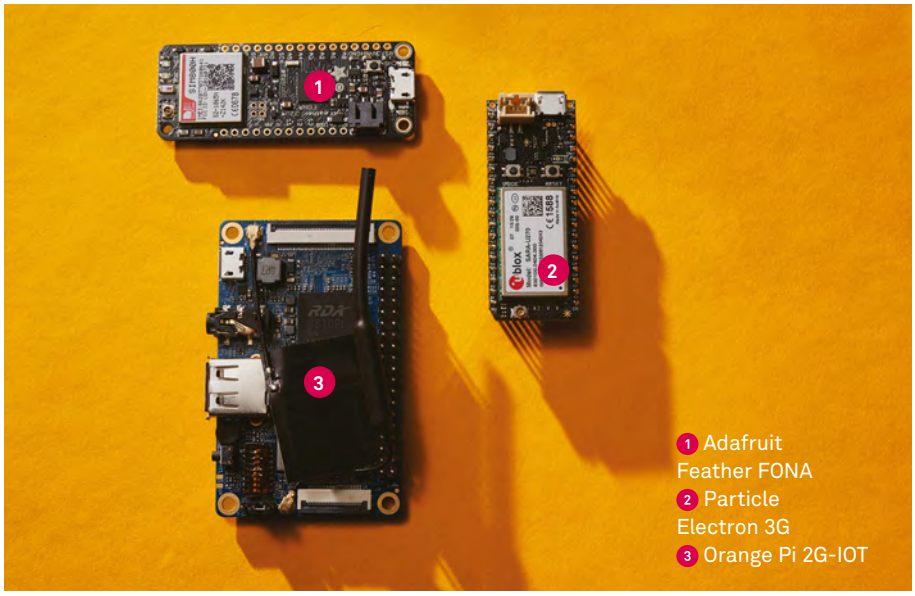
Adafruit's Feather FONA (£50) is a 2G-enabled version of its Feather range of Arduino-like modules. As with most similar products you'll need to add a SIM card, an external antenna and a battery, and house the whole lot in a durable case for true off-the-grid applications. It's far more versatile than a cheap mobile.

Particle Electron 3G

The Particle Electron 3D (around £52) is a small, 51mmx32mm module with an ARM Cortex M3-based microcontroller plus 128K RAM and a massive 1M flash memory, and 3G connectivity. Unlike other cellular cards, it comes with its own SIM card and data plan, together with a plug-in system for extensions.

IoT BOARDS

The Internet of Things (IoT) is a major growth area and there's no shortage of SBCs to fill this need



1 Adafruit Feather FONA
2 Particle Electron 3G
3 Orange Pi 2G-IOT

The Internet of Things (IoT) comprises 'things', as opposed to computers, that are connected to the internet and are commonly accessible via a web interface. These range from sophisticated pieces of equipment such as home appliances to simple sensors that keep an eye on the environment. In this latter category, IoT sensors have been used to implement earthquake early-warning systems, for example.

Many of the SBCs on the market are positioned at IoT applications, which raises the question of how an IoT board differs from any other SBC. Given the wide range of applications, there's no single answer. From the phenomenal growth forecasts for the IoT, it's clear that, for some applications, low cost is essential and many of the cheaper Arduino products are eminently suitable for this.

For other applications, long-term unattended operation is a must, so low power consumption – perhaps coupled with the ability for the board to enter a deep sleep mode between periods of activity – is essential. Again, many Arduino-like boards offer this important feature. Here, however, we're concentrating on a specific feature that will enable IoT devices to be set up in remote locations, a key feature for many environmental-sensing applications. Most of the Raspberry Pi-like boards have Ethernet ports with wireless interfaces such as Bluetooth and Wi-Fi, and these can be provided as add-ons for Arduino-like boards. For equipment located in buildings with Wi-Fi or a wired broadband connection, this provides access to the internet.

In remote areas, however, we need boards that can connect to the net via mobile phone networks. Using a separate mobile phone would not meet either the cost or the power requirements, but SBCs are available with the radio circuitry of a mobile built in. High speed isn't a requirement for most IoT applications so there's no need to go for the latest 4G design – most boards offer either 3G or the older 2G (GSM). Do make sure there are no plans to shut down the 2G networks in your country before investing, though!

QUICK TIP

Can't get a signal?

Before embarking on a project involving IoT devices in remote areas, check for network coverage. Mobile phone network coverage often drops off rapidly in many rural areas (and even not-so-rural ones).

Get 6 issues FREE

When you subscribe*

FREE DVD! LATEST LINUX MINT!
www.linuxuser.co.uk
Linux User & Developer
THE ESSENTIAL MAGAZINE FOR THE GNU GENERATION
SPECIAL REPORT
MICROSOFT
How I learned to stop worrying and love Linux
Is Microsoft a friend or foe?
INSTALL 2 RASPBERRY PI DISTROS TODAY!
Pi-optimised Raspbian Stretch Lite and DietPi v150
LINUX MINT
CINNAMON 18.3
Windows-like desktop OS for the Linux user
ZORIN OS 12.2 CORE
43 PAGES OF GUIDES
Security: sniff out reliable exploits
Master GNU Make
Java: Add AI
HUGE ROUNDUP
Get the perfect single board computer
21 tiny boards on show
ALSO INSIDE
FREE DVD INCLUDES LINUX MINT
ULTIMATE DISTROS & APPS
Project Kubic
Arduino is back!
Pure hardware
46 PAGES OF EXPERT GUIDES
FREE DVD! 4.7GB OF DISTROS & TOOLS
100 WAYS TO MASTER THE TERMINAL
Become a command line guru in 24 hours
ZeroPhone
The hero building a $50 phone with the Pi Zero
Top pen-testing techniques
49 EXPERT GUIDES
DVD UBUNTU 17.10 SPECIAL
MASTER UBUNTU 17.10
NEW RELEASE
Install it Customise it Master it
FREE DVD TRY 2 UBUNTU SPINS
ULTIMATE RESCUE & REPAIR KIT
Vivaldi

"The open source authority for enthusiasts and developers"

FREE resource downloads in every issue

Offer expires 28 February 2018

Order hotline **+44 344 848 2852**

Online at **www.myfavouritemagazines.co.uk/sublud**

*This is a US subscription offer. 6 free issues refers to the USA newsstand price of $16.99 for 13 issues being $220.87, compared with $112.23 for a subscription. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Payment is non-refundable after the 14 day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: bit.ly/magtandc. Offer ends 28 February 2018.

techradar.

The source for tech buying advice

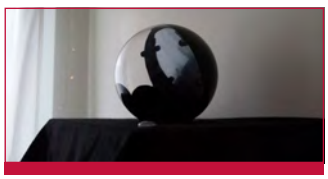
techradar.com

THE ESSENTIAL GUIDE FOR **CODERS & MAKERS**

PRACTICAL Raspberry Pi

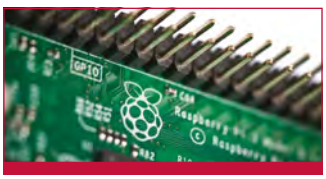
70 "A deceptively simple way to display the weather forecast"

Contents



70 Pi Project: a 'crystal ball' for weather

72 Mod Minecraft on the Pi using Python



74 Access GPIO pins with the GPIO Zero library

78 Using your RDBMS with Python

Jenny Hanell is a Masters student in Human Computer Interaction from Stockholm, Sweden. Beside her studies, she likes to develop Android apps, as well as hiking in the forest with her dog. Jenny is very interested in the psychological part of HCI and how technology affects our behaviour.

Like it?

Watch the Sphaera in action on Jenny Hanell's YouTube channel **https://youtu.be/4d07RXYLsJE**

Further reading

To learn how to make your own Sphaera head to **https://www.instructables.com/id/SPHAERA**. You can download the video files for the holograms at: **http://bit.ly/SphaeraVideoFiles** and the code here: **http://bit.ly/SpaeraGitHub**.

Sphaera weather forecaster

Weather forecasting gains a magical edge with a globe that predicts the weather by a touch of the hand

Sphaera is a deceptively simple way to display the weather forecast for the next twelve hours (using the OpenWeatherMap API) by touching five photoresistors positioned evenly around a crystal ball. The design of Sphaera is meant to blend in with a home environment – it's something you might expect to see in the hallway that you can interact with before leaving the house.

What was the inspiration behind Sphaera?
Sphaera was made by myself and two friends. We were particularly inspired by the ideal of a crystal ball and the basic idea of looking into the future. We spent some time figuring out what kind of future events we wanted to visualise, and realised that the weather was a perfect match, since it's easy to fetch from open APIs and it would look nice to project different weather states as video holograms. Since we really liked the feeling of the glass surface, we decided to not place any sensors or other material on the ball itself, but instead put everything inside.

What was your approach to the interactive design?
We wanted the interaction with the ball to be as close as possible to how you interact with a classic 'crystal ball', which means placing your hands above or around the ball in order to interact with it. This was intended to give the user an idea of how to use the device without any instructions, but it turned out that it was quite difficult to achieve. Even if the user covers one of the light sensors, it is difficult to implement what actually happens without ruining the design or the magical feeling.

> "The Raspberry Pi had everything we needed"

So what was your solution?
We tried many different designs. Since the holograms require darkness in order to be visible, we started out by giving the glass globe a dark hood, with the sensors hidden inside the fabric. This made it look like a little character from The Lord of the Rings or some other fantasy story, and even though it was quite amusing, that kind of ruined the elegant feeling that we wanted. So we decided to iterate a little bit, get rid of the hood and paint the inside black to achieve the required darkness.

What was the most challenging part of the project?
Definitely working on the aesthetic details. This involved placing all the sensors inside the globe and keeping them them in place using strong glue; pulling down the conductive threads without having them touch each other; painting half of the globe black on the inside; and finally placing a piece of plastic inside the globe at the perfect tilted position, to reflect the holograms.

What made you decide to use the Raspberry Pi?
Since we were all already familiar with Arduino, we wanted to try something new. Also, the Raspberry Pi had everything we needed for this project, such as built-in Wi-Fi and Bluetooth. It was also nice that we could save some money and re-use the screen to play the hologram videos instead of buying a new one. However, that made the whole device slightly large and clumsy, and a smaller screen would probably be better if anyone else wants to try the project.

What would you do differently in hindsight?
Use less attractive but more practical solutions. For example, we used a thin and almost invisible conductive thread that went from each sensor's legs down to the breadboard. The reason for that was to avoid any ugly visible wires, but it was actually very difficult to attach it to the sensors and to make sure we didn't break the thread's conductivity while gluing and painting it. A normal slim black wire would probably work fine.

Do you have any features you want to add?
It would be nice to visualise more weather data such as temperature, and also to show the weather forecast for the whole week, not just for the next 12 hours. It would also be cool to add speech recognition so that you can ask the globe about the weather. And of course more sensitivity – right now it only projects the standard weather conditions, but it would be nice to have different kinds of rain states, for example.

What kind of projects interest you?
I like projects that are relatively simple to start working on and don't require too much knowledge. I like to learn by doing and to jump into new projects of different character. But I really enjoy working with visualisations in different forms – to take data that is already available, put it in a new context and make it attractive and interesting somehow.

What are you hoping to do next?
My next project will probably be something related to my toddler's room. I have some ideas about an interactive lullaby lamp or some kind of playful painting with moving animals. I'm also curious to use the Raspberry Pi in combination with Google's Android Things – it looks fun!

Components list

- Raspberry Pi 3 (model B plus keyboard, mouse and microSD card
- Glass globe
- A round piece of soft plastic, size depending on the diameter of the globe
- Fabric (about 1mx1m)
- LCD screen, HDMI cable and adaptor (DVI/VGA)
- 5 CdS photocells
- 4 1uf capacitors
- 1 push button
- Breadboard, chords and heat-shrink tubes
- Conductive thread (10m)
- 9 pieces of black sponge (2x1cm)
- A cardboard box big enough to fit the screen
- Scissors
- Cellplast
- Bluetooth speaker

Touch time

The five photoresistors each cover a different time period. For example, the current weather is projected when you cover the first one alone, while the forecast for 12 hours' time is projected when covering all five photoresistors, as each one adds three hours to the forecast.

Precision plastic

The plastic that helps to create the floating hologram effects was carefully fixed in place with four pieces of sponge around each edge. Getting the position right required some experimentation by projecting a hologram. Jenny recommends putting the screen in a tilted position to help with determining this.

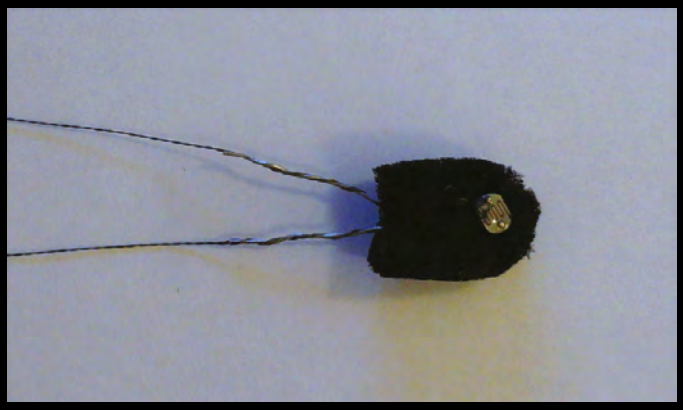
Dark view

The weather is projected as a floating hologram inside the glass globe, which is made visible by painting half of the globe pure black. This was also used to prevent the user seeing the plastic placed inside. There's also a black border on the front lower part of the ball so the user can't see the LCD screen.

Wiring the weather

The holograms are created by projecting graphics through a hole created beneath the crystal ball onto a carefully positioned and tilted piece of plastic. The hologram graphics are custom MP4 files for different types of weather (sunny, cloudy, snowy and so on) that are displayed on an LCD screen, and the sound is pumped out of a Bluetooth speaker.

Above The whole project is controlled by a Raspberry Pi 3, which is connected to both the photoresistors via a breadboard and to the LCD screen. The Pi runs Python 3 code (which you can find at **http://bit.ly/SpaeraGitHub**) that accesses the OpenWeatherMAP API via Wi-Fi and determines the relevant weather MP4 file to play on the screen. This is then displayed inside the ball as a 'floating' animation.



Above Each photoresistor was placed inside a black sponge, with the top pointing upwards and the legs horizontal towards one of the short sides. Conductive thread was cut and wired around each photoresistor leg, which was glued inside the glass ball. Hanell describes the whole process as very difficult because the thread was so thin and she had to ensure the threads were spread out but didn't touch. She also faced problems with conductivity.

Calvin Robinson is Head of Computer Science at an all-through state school in Barnet. Calvin also consults with schools all over London, providing high-quality computing curricula.

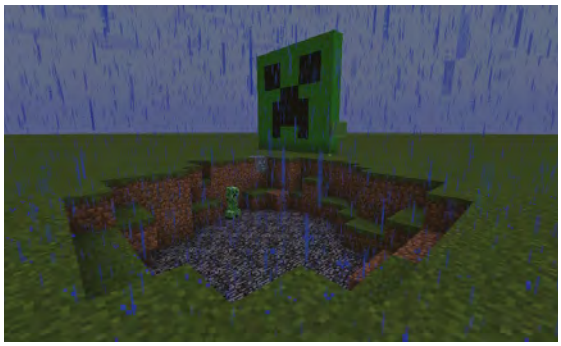
Resources

McPiFoMo
http://rogerthat.co.uk/McPiFoMo.rar

Block IDs
http://minecraft-ids.grahamedgecombe.com

Mod Minecraft on the Raspberry Pi using Python

In the final part of our Minecraft Rasperry Pi series, discover how to use Python to mod and tweak Minecraft



This tutorial is written with Minecraft Pi Edition in mind, but you don't have to be running Minecraft on a Raspberry Pi to follow along. We've put together a little package that will work on any version of Minecraft, so if you'd like to run this tutorial on your favourite flavour of desktop Linux – Pi or no Pi – you can. To allow Python to hook into Minecraft you'll need to install McPiFoMo by extracting the contents of the **.minecraft** directory into your **~/home/.minecraft**. McPiFoMo includes MCPiPy from MCPiPy.com and Raspberry Jam, developed by Alexander Pruss. Provided you have Python installed, which of course is pretty standard on most distros, no additional software is required, other than your favourite text editor or Python IDLE.

Python scripts in this tutorial should always be saved in **~/home/.minecraft/mcpipy/** regardless of whether you're running Minecraft Pi Edition or Linux Minecraft. Be sure to run Minecraft with the 'Forge 1.8' profile, included in McPiFoMo, for your scripts to run correctly.

This is the final chapter of the Minecraft Pi series of tutorials in Python. See issues 178-186 for more – if you missed an instalment, you can visit **https://www.myfavouritemagazines.co.uk** to buy back issues.

01 Jump boost

Give your Minecraft player character a boosted jump by altering the Y factor to taste:

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()
playerPos = mc.player.getPos()
playerPos.y = playerPos.y + 10
mc.player.setPos(playerPos.x,playerPos.y,playerPos.z)
```

02 Auto-build

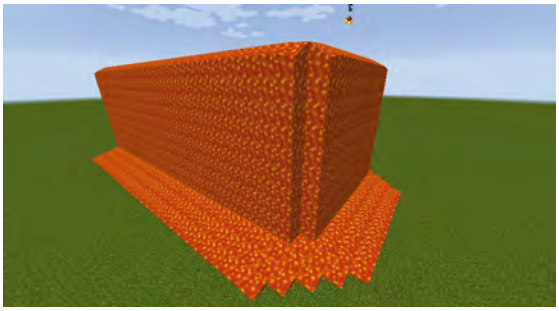
Give Minecraft a set of coordinates and the `setBlocks` command can fill in the gaps. Adding to the previous code:

```
height = 10
width = 10
depth = 10
blockID = 10
mc.setBlocks(playerPos.x, playerPos.y, playerPos.z, playerPos.y + height, playerPos.x + width, playerPos.z + depth, blockID)
```

03 Setting a custom block ID for auto-build

Instead of hard-coding the blockID we can create a 'user input' to request one from the player. Working with the code from steps 1 and 2, this method would mean commenting out the original `blockID` variable and replacing with the one below:

```
blockID = input("Enter your blockID: ")
mc.setBlock(playerPos.x,playerPos.y, playerPos.z, blockID)
```



04 Activating Immutable mode

While we're experimenting with the auto-build tool we may want to stop ourselves and/or other players from destroying blocks. We can do this by setting Immutable mode, which makes blocks indestructible.

```
mc.setting('world_immutable', True)
mc.postToChat("Immutable mode activated")
```

You may also want to add an option to set this Boolean to False, to deactivate Immutable mode.

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

mc.setting('world_immutable', False)
```

05 Glitchy signposts

For a glitchy sign, following on from the above code with **playerPos** and **blockID** variables initialised, you'll need to insert **import time** at the top of your code and then include:

```
states = [0, 5, 6, 11, 3, 8, 9, 2, 12, 7, 4, 16]
for state in states:
mc.setBlock(playerPos.x, playerPos.y, playerPos.z, blockID, state)
time.sleep(0.3)
```

06 Auto-foliage: making trees

Automatically building blocks is fine, but what if you want to automatically build an entire forest?

```
def makeTree(x,y,z):
wood = 10
leaf = 10
nothing = 0
mc.setBlocks(x,y,z,x,y+3,z,wood)
mc.setBlocks(x-2,y+4,z-2,x+2,y+5,z+2,leaf)
mc.setBlocks(x-1,y+6,z-1,x+1,y+7,z+1,leaf)
mc.setBlock(x-1,y+7,z-1,nothing)
mc.setBlock(x+2,y+5,z-2,nothing)
mc.setBlock(x+2,y+4,z+2,nothing)
mc.setBlocks(x+1,y+6,z-1,x+1,y+7,z-1,nothing)
mc.setBlocks(x+1,y+6,z+1,x+1,y+7,z+1,nothing)
mc.setBlocks(x-1,y+6,z+1,x-1,y+7,z+1,nothing)
```

In this block of code we're simply creating a new function called **makeTree** with three passable integers for the coordinates.

> "Automatically building blocks is fine, but what if you want to automatically build an entire forest?"

07 Auto-foliage: spawning the trees

Now we need to spawn the trees. Change the integer to make them more/less dispersed:

```
makeTree(x+1,y,z)
makeTree(x+10,y,z)
makeTree(x+10,y,z)
makeTree(x+1,y,z+10)
makeTree(x+10,y,z+10)
makeTree(x+10,y,z+10)
```

08 Hijacking another player's screen, part 1

List all the player IDs in chat:

```
players = mc.getPlayerEntityIds()
mc.postToChat(players)
```

Find the ID of the player whose screen you'd like to observe (or rather, hijack) and we'll use that number as an integer in the following step. Be sure to have permission from the player(s) whose screen you're about to take over beforehand, or there may be trouble...

09 Hijacking another player's screen, part 2

To take over another player's screen, we use the list created previously, and set our camera to follow that player:

```
mc.camera.setFollow(players[1])
```

Switch back to your camera at any time with:

```
mc.camera.setFollow(players[1])
```

Python for Minecraft Pi

Using Python we can hook directly into Minecraft Pi on the Raspberry Pi to perform complex calculations, alter the location of our player character, spawn blocks into the game world to create all kinds of creations – both 2D and 3D – and read/write pre-written scripts from text files to create our own authentic looking Non-Player Characters. We can program pretty much anything from pixel-art to chat scripts that communicate directly with the player. In this issue we wrap up the Minecraft Pi series by modding our player character, glitch some signs and hack a friend's screen.

With each issue of **LU&D** we've taken a deeper look into coding Python for Minecraft Pi, with the aims of both improving our Python programming skills and gaining a better understanding of what goes on underneath the hood of everyone's favourite voxel-based video game. We hope you enjoyed the ride!

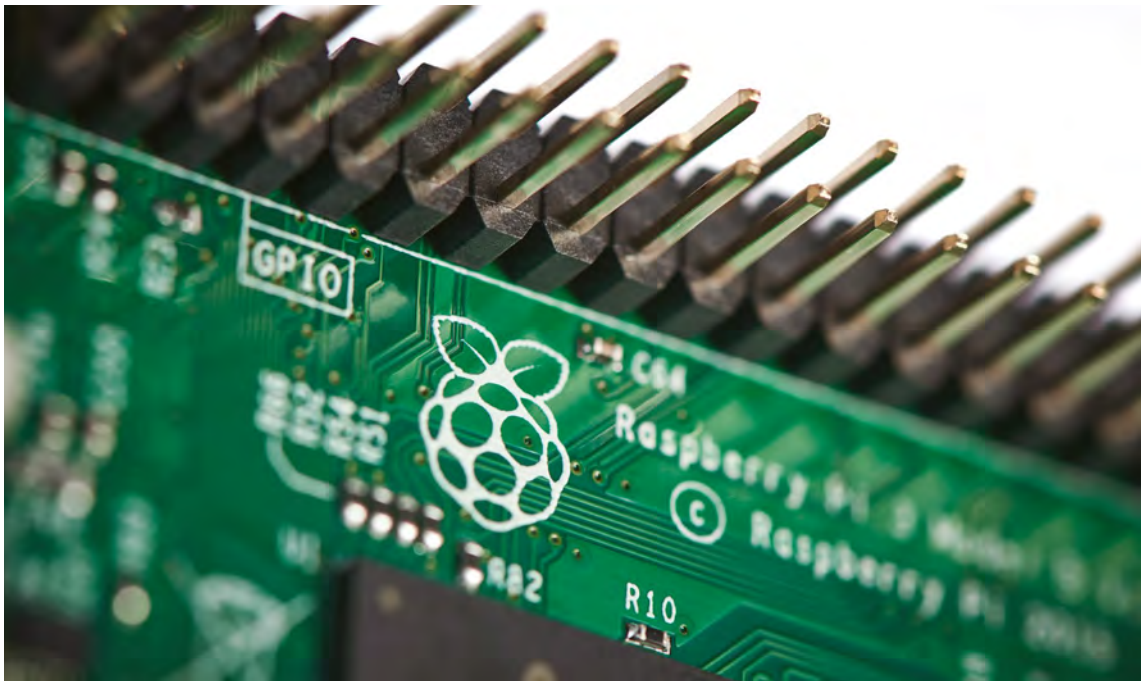
Dan Aldred is a Raspberry Pi enthusiast, teacher and coder who enjoys creating new projects and hacks to inspire others to start learning. He's currently working with the Raspberry Pi Google Home Assistant.

Resources

- Raspberry Pi
- LED
- Push button
- Jumper wires (female to female)

Unlock the potential of GPIO pins with the GPIO Zero library

Make using the GPIO pins easy, fun and expand your interaction with a wide range of components and sensors



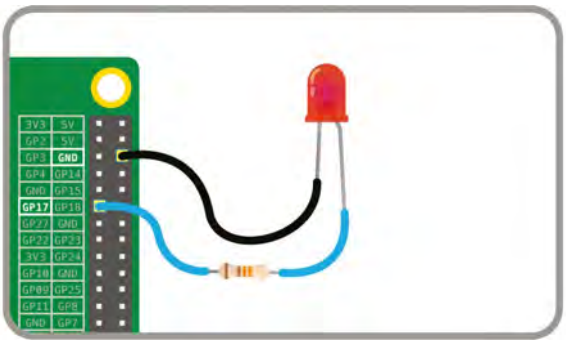
Most of us will probably remember the first time we wrote a program to light up an LED, control a motor or took a temperature reading. Nowadays this hardware and these components are relatively cheap – you can purchase a suite of sensors for the equivalent of the original cost of buying one. Components and sensors have shrunk in size with many now being etched into small add-on boards.

The libraries to control the hardware has also evolved and with this change comes the amazing GPIO Zero Python library, created by Ben Nuttall and Dave Jones, which provides a simple interface to the Raspberry Pi GPIO pins. Consider the basic example of making an LED blink: a traditional RPi.GPIO program requires 16 lines of code. With GPIO Zero you can achieve the same outcome with only five.

But don't be fooled into thinking that GPIO Zero is dumbed-down coding; GPIO Zero supports a wide range of components, sensors and parts. In this tutorial we will begin by trying out a few basic recipes before moving onto the internal devices. We'll also create a simple app to control an LED via your phone or tablet.

01 Wire up the LED

Let's introduce the GPIO Zero library with a basic LED control program. As the tutorial progresses and we introduce more complex features we'll use the LED as the responding action, so this will come in handy. Start by wrapping a suitable resistor around the positive leg of the LED and then attach the jumper wires to the two legs. The positive wire connects to GPIO pin 17 and the other wire to a ground (GND) pin.



Raspberry Pi Foundation (raspberrypi.org)

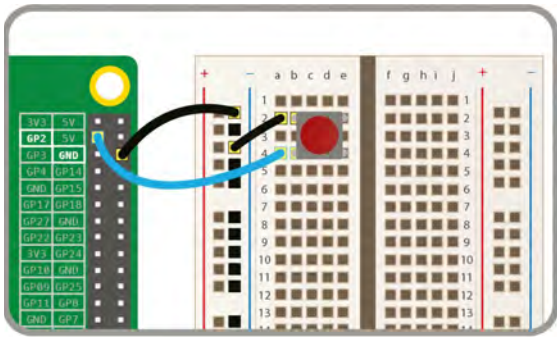
02 Flash that LED

Now let's write simple code to flash the LED on and off. Import 'LED' from the GPIO library and assign the GPIO pin where the LED is attached – in this case, GPIO 17. Then set up a `while` loop to turn the LED on for one second and then off. Save the program and run it. How easy is that?

```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```



Raspberry Pi Foundation (raspberrypi.org)

03 Wire up the button

Now let's wire up a button. Take the breadboard and attach the button into the holes, leaving space to connect the wires. Connect the first wire (blue) to GPIO pin 2; this provides the current for the circuit. Attach the other end to the top leg of the button. The second wire connects to a ground (GND) pin and completes the circuit. This example uses physical pin number six, but any of the other GND pins are suitable.

04 Code the button presses

This is just as easy – it's a simple program to control the reaction to the button being pressed. There are several 'button' programs. Our first example assigns the GPIO pin for the button on line two, then waits for a button to be pressed on line three and when pressed prints a message, on line four. In the second example, we create a function which is then called each time the button is pressed. This enables us to create more complex outcomes and actions.

```
from gpiozero import Button
button = Button(2)
button.wait_for_press()
print("Button was pressed")
```

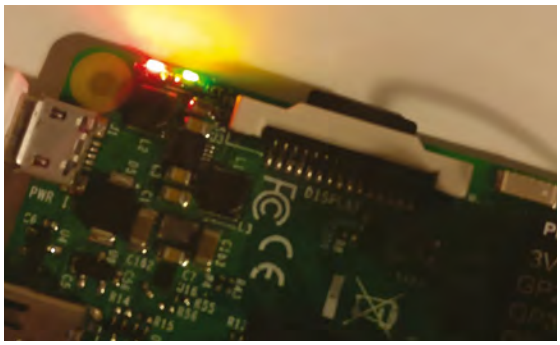
Or somewhat more usefully:

```
from gpiozero import Button
from signal import pause

def say_hello():
    print("Hello!")

button = Button(2)

button.when_pressed = say_hello
pause()
```



05 Control the Pi's own LEDs

The GPIO Zero library makes it possible to interact and control the Raspberry Pi's built-in LEDs. These are the red power status and the green activity indicators located near the power port. Before creating your program, you need to configure each LED. Open the LX Terminal and type the following command: `echo none | sudo tee /sys/class/leds/led0/trigger` then press Enter. Next configure the second LED using the command `echo gpio | sudo tee /sys/class/leds/led1/trigger`. Now open your Python file and create the program to control them.

06 Write the program

This is another simple bit of code that flashes the activity LED followed by the power LED, then pauses.

```
from gpiozero import LED
from signal import pause

power = LED(35)
activity = LED(47)

activity.blink()
power.blink()
pause()
```

07 Reset the LEDs

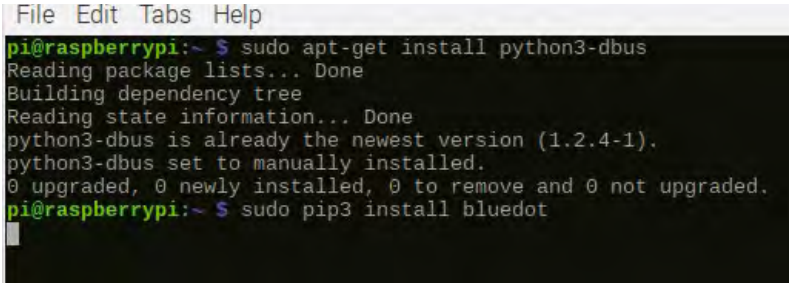
If you need to return the LEDs back to their orginal state – to use them for their original power and activity indication purposes – in the Terminal window type `echo none | sudo tee /sys/class/leds/led0/trigger` and then `echo gpio | sudo tee /sys/class/leds/led1/trigger`. Once you have completed this, reboot your Raspberry Pi by typing `sudo reboot` and then press Enter.

08 Use the pin-out tool

When using the GPIO pins it is essential to have a pin reference guide to hand. Many of us will have a scrap of paper, a neat poster or a website for this. GPIO Zero comes installed with an extremely handy board diagram utility. Open the Terminal window, type `pin out` and you will be presented with a graphical layout diagram. It also provides additional details about the status of the Wi-Fi, Bluetooth and other ports.

09 Install Blue Dot

Blue Dot is a super-simple app which enables you to interact with LEDs, motors and other components via a large blue dot on your phone or device. In this example we will use it to control the LED from step one. To start, open the LX Terminal window and install the Blue Dot software: type `sudo apt-get install python3-dbus` and then `sudo pip3 install bluedot`. To upgrade and add additional features, use the command `sudo pip3 install bluedot upgrade`.

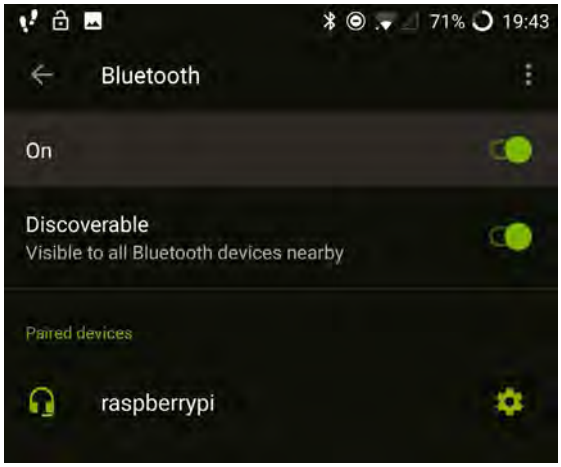

File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get install python3-dbus
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-dbus is already the newest version (1.2.4-1).
python3-dbus set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo pip3 install bluedot

10 Download Android Blue Dot app

While the Python library is installing, head over to the Android Play Store and install the Blue Dot app. Next, turn on your phone's Bluetooth and ensure that the discovery mode is set to Discoverable. This ensures that the Raspberry Pi can locate your device.

11 Pair your Pi and device

Now to pair your device with your Raspberry Pi. Locate the Bluetooth symbol at the top-right of the desktop. Right click the symbol and select 'Turn on' and 'Make discoverable'. You should see your device listed: select it and connect to it. You may be required to enter a shared pin number, depending on which device you're using. An alternative method to connect is to pair via your phone; search for nearby devices and then select your Raspberry Pi from the list.



12 Create your program

Start a new Python file and add import the Blue Dot module (line one) and LED (line two). Next we create variables to hold the Blue Dot commands and to hold the GPIO number of the LED. On line five, we use a `while` loop, in which we check if the Blue Dot button on your app has been pressed, line six. If it has, we turn the LED on. When it is released turn the LED off – line nine. Ensure that your Pi and device are still paired and then run your program.

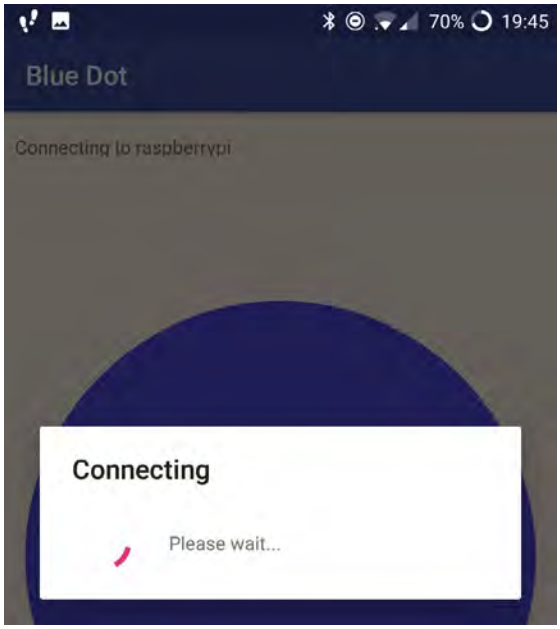
```
from bluedot import BlueDot
from gpiozero import LED

bd = BlueDot()
led = LED(17)

while True:
    bd.wait_for_press()
    led.on()
    bd.wait_for_release()
    led.off()
```

13 Run the program

With the program running and the Pi and phone paired, open the Blue Dot app on your device. From the list select your Pi and establish a connection; you'll see a confirmation message. To turn the LED on press the dot; when you release it the LED will turn off. You can use the



dot to control other outputs such as motors and buzzers – check out the GPIO Zero website for more recipes.

14 Ping a server

In this example we'll ping a server and check its status as online or offline. Using the same wiring as step one, in the Python file we import PingServer and LED on line one, and 'pause' on line two. Replace `google.com` with another address on line four. On line five, we add a 60-second delay between checks and then if the server is online, we turn the led on, line six. If the LED goes off, you know the site or server is down.

```
from gpiozero import PingServer, LED
from signal import pause
led = LED(17)

google = PingServer('google.com')

led.source_delay = 60  # check once per minute
led.source = google.values

pause()
```

15 Check Pi CPU temperature

GPIO Zero's CPU Temperature module enables you to use an LED as a simple warning light when the Pi's CPU reaches a certain temperature. Import the required CPUTemperature, LED, pause and time modules, line one. Create a `while` loop and query the temperature, line six – you can set the values to check between a specific range. On line nine we use an `if` statement to see if the CPU temperature is greater than 60 degrees; a higher temperature will result in the LED being turned on. Alter the values as you like and run your program.

```
from gpiozero import CPUTemperature, LED
from signal import pause
import time

hot = LED(17)

while True:
    cpu = CPUTemperature(min_temp=50, max_temp=90)
    print('Initial temperature: {} C'.format(cpu.temperature))
    time.sleep(1)

    if cpu.temperature > 60:
        hot.on()
    else:
        print ("Cool!")
```

16 React to time of day

This final program takes a time reading from the Pi's internal clock and responds by turning on the LED when the time is within a specific range. First we import the TimeOfDay and LED modules, line one, then the time, line two, and again 'pause' on line three. Set your specific time range on line five – in this example between the hours of 7am and 9am. You may need to adjust these values if you are testing after 9am. Finally, set the LED to turn on when the time meets the morning values, between 7am and 9am.

```
from gpiozero import TimeOfDay, LED
from datetime import time
from signal import pause

light = LED(17)

morning = TimeOfDay(time(7), time(9))
light.source = morning.values

pause()
```

Remote GPIO

The GPIO Zero library also includes a feature that enables you to control the GPIO pins from other devices. One of the pin libraries supported, `pigpio`, provides the ability to control pins remotely over the network. This means that you use GPIO Zero to control devices connected to a Raspberry Pi on the network. You can trigger the LED on and off from another computer, turn on a motor or return the reading from a sensor. You can do also do this from Linux and even from a PC. Check out more details here: **https://gpiozero.readthedocs.io/en/stable/recipes_remote_gpio.html**

Using your RDBMS with Python

When you have a lot of data to work with, you will likely need to use a RDBMS – so this month, learn how to use one with Python

Joey Bernard is a true renaissance man. He splits his time between building furniture, helping researchers with scientific computing problems and writing Android apps.

Why Python?

It's the official language of the Raspberry Pi. Read the docs at **python.org/doc**

Last issue, we looked at how to use SQLite to use data without needing a full RDBMS (Relational DataBase Management System). This is fine when you have a limited amount of data, but at some point you'll need the extra performance. Options such as MySQL or Postgresql are available that focus on providing your data as efficiently as possible. We won't be looking at how to set up or manage the database; instead, we'll assume that there is already an existing database and focus on how to use it with Python. Also, we'll be using MySQL as the example RDBMS; the concepts are very similar from one database to another with only the syntax changing much. To install the Python module for Debian-based distributions, such as Raspbian, use `sudo apt-get install python-mysql.connector`. If you are using Python 3.x, you can replace `python-mysql.connector` with `python3-mysql.connector`.

> "We should manage connection errors"

Once the Python module is installed, import it into your program with `import mysql.connector`. The first step is to connect to the MySQL server. The basic form is:

```
my_conn = mysql.connector.connect(user='username', password='password',
  host='127.0.0.1', port='3306',
  database='mydb')
```

In this example, the MySQL service is running on the local machine, hence the host being set to `127.0.0.1`. If it is running on another machine, you can set the host parameter to the relevant IP address or hostname. If it's listening on the default port, 3306, you can leave it off the parameter list. In previous articles, we haven't worried about exception handling, but when it comes to connecting to database, we should look at how to manage possible connection errors:

```
from mysql.connector import errorcode
try:
  my_conn = mysql.connector.connect(user='username', database='test1')
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password")
  elif err.errno == errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
  else:
    print(err)
else:
  my_conn.close()
```

This all assumes that the database already exists on the MySQL server. If it doesn't, you can leave the database parameter out of your connect call, and create the database after connecting to the server. The following code would create a new test1 database:

```
DB_NAME = 'test1'
my_conn = mysql.connector.connect(user='username')
my_cursor = my_conn.cursor()
my_cursor.execute("CREATE DATABASE {} DEFAULT CHARACTER SET 'utf8'".format(DB_NAME))
my_conn.database = DB_NAME
```

This way, you can have your program bootstrap the entire data storage step, assuming that the username you are using has the privileges needed to create a new database.

Continuing the setup, you may need to create tables to store your data before doing any work with it. Just as with creating a database, you will need to have a cursor that can execute SQL statements. The following code will create a table to store names and phone numbers:

```
table_stmt = "CREATE TABLE 'phones' ('name' varchar(50) NOT NULL, 'number' int(9) NOT NULL) ENGINE=InnoDB"
my_cursor.execute(table_stmt)
```

As you can see, we are just handing in SQL statements to be processed by the MySQL server. These types of statements are called DDL (Data Definition Language) statements and you can send in pretty much anything that the MySQL server will understand.

Once your database has been created and properly structured, you need to load data in order to start using it. If you have large amounts of data, you will likely want to bulk-load it directly using the utilities that come with MySQL. If you are loading data as it is being collected within your program, use:

```
add_phone = "INSERT INTO phones (name, number) VALUES (%s, %s)"
phone_data = ('Joey Bernard', 5551234567)
my_cursor.execute(add_phone, phone_data)
```

As you can see, we separated out the insertion statement from the data being inserted. This way, you can easily reuse the add statement. Also, since the data is separated out,

you can more easily do pre-processing to ensure that the incoming data is sanitised. One of the key structures of a RDBMS is the relational part. This means that you may need the row ID of the most recent insertion to use as a key linking it to some other entry in another table. This would look like `row_id = my_cursor.lastrowid`.

By default, connections to the database have autocommit turned off. This means that everything you do is handled through transactions. To ensure that the data change you just made is pushed to the database, you need to commit the transaction with `my_cursor.commit()`.

This will commit everything that has happened since the last commit call. This means that you can also rollback changes using the `rollback()` method of the cursor object. Again, this applies to everything that has happened since the last commit.

Once you have a database that's fully loaded with data, how do you pull it back out in order to work with it? You can hand in a `SELECT` statement, using the `execute()` method of the cursor object. As with the `INSERT` statement above, you can separate the statement from any search parameters that you want to use to constrain your query. For example, the following code will pull up all of the data in the test1 table:

```
my_cursor.execute("SELECT * FROM test1")
```

There are two ways to pull out the results from this query. If you want to pull out one of them, you can use the `fetchone()` method of the cursor object. This will give you a tuple containing the next row in the list of rows returned by your query. There are also `fetchmany()` and `fetchall()` methods that allow you to grab larger chunks of returned data. If you wanted to step each returned row and do something with each one, you can use:

```
for (name,number) in my_cursor:
  print("Name: {}, Phone number: {}".format(name, number))
```

This works because the cursor object can be used as an iterator.

As users work with your program, you will need to alter data stored in your MYSQL database. If you need to update stored information, you can use the `UPDATE SQL` statement. The code below will update my phone number:

```
my_cursor.execute("UPDATE test1 SET number=5559876543 WHERE name='Joey Bernard'")
```

If you find that you need to clean up old data, you can remove it with:

```
my_cursor.execute("DELETE FROM test1 WHERE name='Joey Bernard'")
```

When your data collection gets large enough, you may want to take advantage of the strengths of an RDBMS by creating and using stored procedures within the MySQL database. We'll assume that you have already created a stored procedure within the database, named `my_func`. You can then use the `callproc()` method of the cursor object:

```
my_cursor.callproc('my_func')
for result in my_cursor.stored_results():
  print(result.fetchall())
```

You need to use the `stored_results()` method to pull each result out and then use its `fetchall()` method to get the actual returned data. When you are done, don't forget to clean up after yourself with:

```
my_cursor.close()
my_conn.close()
```

And now you are ready to handle even larger amounts of data!

What about PostgreSQL?

While MySQL is very popular, it does have its limits. For more complex data-storage needs, you may decide to use PostgreSQL instead. The most popular option for a Python module to work with a PostgreSQL database is psycopg2. You can install it with `sudo apt-get install python-psycopg2`.

Using this module will look familiar to what we covered with MySQL, with both minor and major syntax variations. For example, you can connect to a database and get a cursor with:

```
import psycopg2
my_conn = psycopg2.connect("dbname=test1 user=username")
my_cursor = my_conn.cursor()
```

As you can see, the main difference to this point is that the parameters for the `connect()` method are named differently and aren't separated by commas. Interacting with the database is handled the same way as with MySQL. That is, you can use the `execute()` method of the cursor to run SQL statements against the database. While SQL is standardised, every RDBMS seems to add its own extensions to the language. This includes MySQL and PostgreSQL, so don't expect to be able to seamlessly move queries from one database to another. This module also uses similar methods to the MySQL module to get results out, namely the methods `fetchone()` and `fetchall()`. You even have the `callproc()` method to execute stored procedures within the database.

As you're probably moving PostgreSQL because you have too much data, you'll likely need to worry about how much data is coming back from queries. By default, the entire set of results comes back in the client cursor object, which may use huge amounts of RAM on the client. If this happens, you can create and use server-side cursors so that the result set stays on the server. This way, you only need to worry about memory for each single row that you are fetching.

If you want to go fully Python, you can even use it on the server side. The PostgreSQL database allows you to use PL/Python to write stored procedures. This means that you will have Python code from the server to the client. You can install it on your database using the following command:

```
createlang plpythonu dbname
```

ON SALE NOW!

Google Pixelbook Your next laptop!
T3
SMARTER LIVING
UPGRADE YOUR SMART HOME
• Next-gen AI speakers tested
• Top picks for home control
• Perfect multi-room audio
6 AIRPOD KILLER EARBUDS
The best true wireless 'phones
NEW!
Cinema-spec 4K upgrades
Theatre-level projection and Dolby Atmos audio at home
MIRRORLESS CAMERA SHOOTOUT
Gorgeous lightweight snappers for taking amazing photos without the DSLR bulk
TESTED!
All-screen OLED Android phones go head-to-head
WIN!
A PORTABLE HI-RES MUSIC PLAYER
EBIKES VS. PEDAL POWER
Man takes on motor in our off-road race

Available at **WHSmith, myfavouritemagazines.co.uk**
or simply search for 'T3' in your device's App Store

SUBSCRIBE TODAY AND SAVE!
www.myfavouritemagazines.co.uk/T3

GET IT ON Google play zinio Available on the App Store

Reviews

81 **Group test** | 86 **Hardware** | 88 **Distro** | 90 **Free software**

GROUP TEST

Raspberry Pi everyday distros

In addition to powering all kinds of embedded projects, you can use the cheeky little computer as an everyday desktop, with the help of these Linux distributions

DietPi

An extremely stripped-down version of Debian Jessie, DietPi ships with just enough operating system (JeOS) to enable you to build and customise your installation from scratch. The distro ships with a handful of very useful custom scripts to help simplify this task.
www.dietpi.com

Fedora Workstation

Starting with Fedora 25, the leading RPM-based distro now supports the ARM and can run on Raspberry Pi 2 and 3. As with the desktop version, there's a server and a minimal edition besides several for the Pi.
https://arm.fedoraproject.org

Raspbian

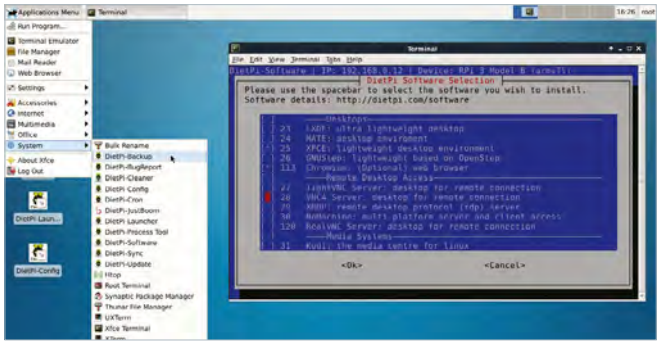
Inarguably the most popular Debian-based distro that's optimised for the Raspberry Pi. The only distro in this group test that's officially supported by the Raspberry Pi Foundation, Raspbian is available both as a standalone download as well as part of the NOOBS installer.
www.raspbian.org

Ubuntu Mate

A tuned build designed for the Raspberry Pi, Ubuntu Mate brings the benefits of the Mate desktop and an Ubuntu base to the ARM architecture platform. The ARM edition can run on both Raspberry Pi 2 and 3 and, like Peach Pi, is based on Ubuntu's 14.04 LTS release.
https://ubuntu-mate.org

DietPi

A minuscule distro that ships with the right kind of tools to flesh it out



Besides the RPi, DietPi is available for a host of single-board computers such as BeagleBone, Banana Pi, Orange Pi and more

Default software

The DietPi distro installs the bare minimum of components you need to flesh out the installation according to your needs. We used its custom apps to install the Xfce desktop environment which automatically pulled in the Firefox and IceWeasel web browsers, along with a couple of Xfce utilities such as the Orage calendar.

Package management

DietPi's secret sauce is its handful of custom scripts, one of which enables you to install software optimised for the Pi. Since it's a Debian-based system, you can use the command-line **apt-get** package management system, or pull-in the Synaptic graphical apps for easier package management.

Desktop and Usability

On first boot you'll have to enable Wi-Fi and then hook it up to a hotspot before the distro can update itself. With that done, you can use the utilities to build your installation. Besides pulling in software, its custom utilities are useful for the usual sort of system-management tasks such as enabling network services and removing unwanted files.

Media playback

In terms of out-of-the-box support, DietPi only includes a configuration utility for the JustBoom Amp HAT audio amplifier. However, once they're installed the browsers can play online videos, and you can install OMXPlayer from the repositories for playing other formats of media.

Overall

A very useful distro for DIYers to build their own installation. Its slew of custom software installation and management scripts also make it accessible to non-technical users.

6

Fedora Workstation

The RaspPi version is as bland and unexciting as its desktop release

Make sure you read through the installation and usage FAQ on the Fedora ARM wiki before you get started with the distro

Default software

The RPi version is an exact copy of the desktop version, with the bare minimum of desktop apps. There's Firefox, LibreOffice, Shotwell, Rhythmbox, Totem, and even the Boxes virtualisation app – which doesn't make sense on the RPi. Like the main desktop release, it uses the latest Gnome 3.26 with the new-look settings window.

Package management

Again, just like its desktop sibling, on the graphical front the distro uses Gnome Software while with the CLI you get the DNF package management system. The repos, however, aren't as fleshed-out as Debian's – so for example you won't find OMXPlayer there and will have to compile it from source.

Desktop and Usability

Like Ubuntu Mate, Fedora launches a first-boot wizard to create users. However, to get the onboard Wi-Fi working you have to manually copy the (non-free) firmware following the instructions on the wiki. The docs also advise you to manually resize the partition to take over the entire SD card. Even after doing all this, the Pi had trouble keeping up with Gnome.

Media playback

The lack of a functional video player in the distro and the repos is a real disappointment. The included Rhythmbox plays music flawlessly, which is piped through HDMI to the TV. You can view videos on YouTube, but they don't play smoothly and aren't really watchable as a result.

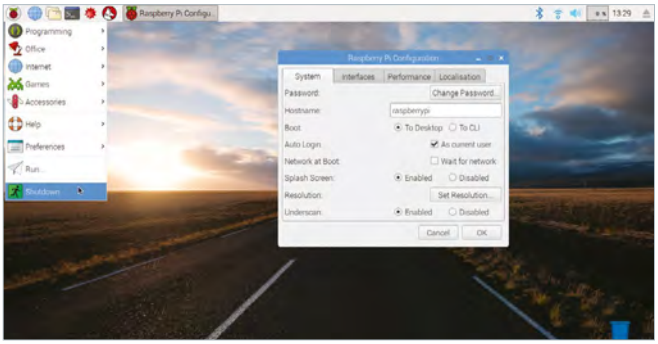
Overall

An exact replica of the Fedora Workstation release for the desktop, the version for the Raspberry Pi 3 comes up short both in terms of included apps and performance compared to its peers.

5

Raspbian

The officially supported distro does everything to deserve that honour



You can now run Raspbian's PIXEL desktop on a regular desktop PC, and it works wonderfully well to resurrect an old workhorse

Default software

The default selection of apps is tailored for those who want to hone their programming skills. There's Sonic Pi, Scratch 2, Minecraft, Sense HAT Emulator, Mathematica, Thonny, Greenfoot, Node-RED and Geany. For regular desktop use there's the Chromium browser with the Flash plug-in, LibreOffice, Claws Mail and VNC Viewer.

Package management

To flesh out the installation, Raspbian uses the no-frills PiPackages app. Besides installing software, the app can also refresh repositories and you can use it to look for and install updates. While it isn't the most attractive looking app store, PiPackages is functional and gets the job done.

Desktop and Usability

After years of sticking with LXDE, Raspbian now has a desktop environment of its own called PIXEL (or 'Pi Improved Xwindow Environment, Lightweight' if you must). It features new icons and artwork to make Raspbian's desktop more appealing. Attention has been paid to aspects such as the window-frame design to make it more modern, as compared to LXDE's rather dated look.

Media playback

While Raspbian doesn't list any multimedia player in the menus, the distro does ship with the CLI-based OMXPlayer. You can also right-click on the volume icon to change the audio output device from HDMI to analog output, which works flawlessly and is a very convenient option.

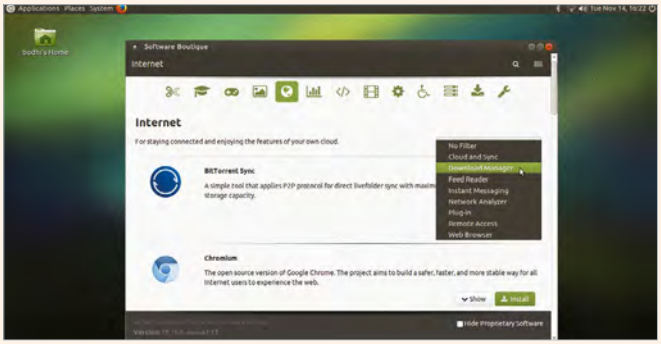
Overall

The official Raspberry Pi distro has solid Debian underpinnings and a good collection of software for its intended audience. It can easily be fleshed-out and has a new look thanks to the PIXEL desktop.

7

Ubuntu Mate

A distro designed for low-powered computers, overflowing with apps



If you can't find the software you're looking for in the app store, you can download one of the popular software centres

Default software

The distro is topped up to the brim with apps. Besides the Raspberry Pi apps that are also in Raspbian such as Scratch 1.4, Minecraft Pi, the IDLE IDE and so on, Mate also has the usual slew of desktop apps including the likes of Pidgin, Thunderbird, Rhythmbox, VLC, Firefox and LibreOffice.

Package management

The distro uses its own app store called Software Boutique that contains a good collection of curated apps in about a dozen categories, such as Education, Graphics, Internet, Games and more. The app store is very intuitive to operate and also gives you the option to hide proprietary apps.

Desktop and Usability

As the name suggests, the distro uses the Gnome2-inspired Mate desktop and is easy to navigate. Mate boots to a first-boot wizard that helps set up the system by creating a user and hooking up the Wi-Fi to a hotspot. First-time users will also appreciate the buttons that enable you to update sources lists, upgrade installed packages in the distro's app store, and fix broken packages.

Media playback

Unlike other aspects of the distro, somewhat surprisingly this one requires some work. The included VLC player is pretty useless in the absence of hardware acceleration, but the CLI-based OMXPlayer works wonderfully well. For better control you can install a graphical front-end such as TBOplayer.

Overall

Ubuntu Mate makes good use of the resources on the Raspberry Pi 3, with fast boot-up times and quick app launches. To top it off, the distro is loaded with all the usual useful desktop apps.

9

In brief: compare and contrast our verdicts

	DietPi		Fedora Workstation		Raspbian		Ubuntu Mate	
Default software	Ships with JeOS & a handful of useful scripts to build a distro from scratch	5	An exact replica of the desktop version with only the most basic of desktop apps included	6	A good collection of apps, especially if you want to hone your programming skills	8	It's liberally overflowing with apps catering for nearly all kinds of users and use cases	9
Package management	Includes a very useful custom script to install software optimised for the Rasperry Pi	7	It's got both graphical and CLI package management, but its repos aren't as fleshed-out as others	6	Uses the graphical PiPackages app and plugs into one of the most expansive repos	7	Its app store contains a curated list of apps and is powered by the Ubuntu repos	9
Desktop and Usability	The set-up process in easy to follow and works without throwing in any surprises	7	Doesn't ship with non-free firmware and makes you run around to get all your devices working	5	The new PIXEL desktop is aesthetically pleasing, and also pleasingly responsive in use	8	The distro uses a first-boot wizard with the lightweight and friendly Mate desktop	9
Media playback	You'll have to install browsers and media players from the repos to play media	5	Plays audio adequately, but lacks a usable video player in the distro as well as in the repos	5	Doesn't list any multimedia player in the menus, but ships with the CLI OMXPlayer	7	Plays multimedia without issues – all you really need is a front-end to OMXPlayer	8
Overall	**A very useful distro for DIYers, but probably more useful for deploying servers**	**6**	**The Workstation release is of little use on the Pi; you're better off trying a lightweight spin instead**	**5**	**Comes with a good collection of apps atop a sparkling new desktop environment**	**7**	**This distro gives the most complete and responsive desktop experience on the RPi**	**9**

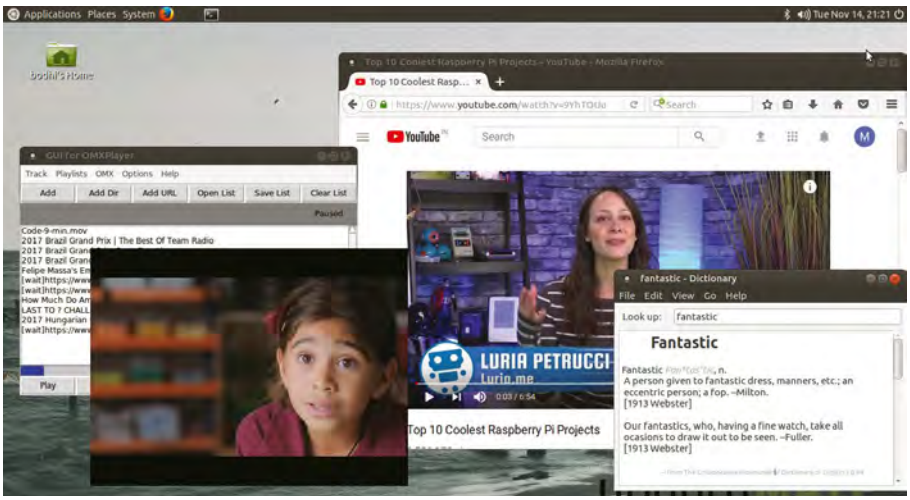
AND THE WINNER IS...

Ubuntu Mate

In addition to those covered here, there are other distros you can use to convert the Raspberry Pi 3 to an everyday desktop. Worth mentioning is SARPi3, which puts Slackware on the RPi. It's a wonderful option for advanced users, but we haven't included it here since it has a more involved installation process that will scare away many first-time users.

Of the ones on test, Fedora Workstation brings up the rear with its lethargic performance. A better option for Fedora fans would be to try one of its lightweight spins, such as the one with the LXQt desktop. DietPi, with its custom scripts for fleshing out the distro, is a wonderful option for the DIYers. But while the scripts do offer the option to install several desktop environments, the distro is more tuned towards helping you deploy all kinds of servers without mucking about with configuration files.

Raspbian has done a commendable job as a desktop distro for the original Pi, which is also why it's the recommended flavour. That said, we'd award this test to Ubuntu Mate.


Applications Places System
Tue Nov 14, 21:21
Top 10 Coolest Raspberry Pi Projects - YouTube - Mozilla Firefox
Top 10 Coolest Rasp...
https://www.youtube.com/watch?v=9YhTQUu
YouTube
Search
GUIfor OMXPlayer
Track Playlists OMX Options Help
Add Add Dir Add URL Open List Save List Clear List
Paused
Play
LURIA PETRUCCI
Luria.me
Top 10 Coolest Raspberry Pi Projects
fantastic - Dictionary
File Edit View Go Help
Look up: fantastic
Fantastic
Fantastic /fanˈtastik/, n.
A person given to fantastic dress, manners, etc.; an eccentric person; a fop. –Milton.
[1913 Webster]
Our fantastics, who, having a fine watch, take all ocasions to draw it out to be seen. –Fuller.
[1913 Webster]

With very little effort, you can use Ubuntu Mate as your standard everyday Raspberry Pi 3 desktop

The distro gives you everything you get with Raspbian plus a lot more. It also required the least amount of tinkering for it to be used as a regular desktop. We had to tweak the RPi's **config.txt** to force the sound through HDMI on one of the boards, and replaced VLC with TBOPlayer, a graphical front-end to OMXPlayer. But that's literally all you need to use Ubuntu Mate on the RPi 3 as your everyday desktop. All things considered, Ubuntu Mate for the RPi is the sincerest attempt to ship a ready-to-use desktop distro for the popular SBC. It doesn't require a trip to the package repository, and in many situations can be put to use straight away.

Mayank Sharma

Future

GIZMODO | UK

Not your average technology website

GIZMODO | UK
UK NEWS GADGETS DESIGN WATCH THIS WTF SCIENCE APPLE ANDROID
Ultimate Smartphone Tech, Form and Design: The GALAXY Note 3 Unpacked
By Your Mobile Life on 11 Nov 2013 at 3:28PM
The Samsung GALAXY Note 3 is the most powerful smartphone on the market today, packed with features from the lightning fast hardware to software apps that have been designed to streamline your life.
The all new TomTom GO
Now with Lifetime TomTom Traffic
LEARN MORE
TOMTOM
GIZMODO | UK
15 DAYS UNTIL CHRISTMAS
MacBook Air

EXPLORE NEW WORLDS OF TECHNOLOGY GADGETS, SCIENCE, DESIGN AND MORE

- Fascinating reports from the bleeding edge of tech
- Innovations, culture and geek culture explored
- Join the UK's leading online tech community

www.gizmodo.co.uk

twitter.com/GizmodoUK facebook.com/GizmodoUK

HARDWARE

Google Pixelbook

The best Chromebook to date – bar none

Price
£999
Website
https://store.google.com
Specs
CPU: 1.2GHz Intel Core i5-7Y57
Display: 12.3-inch QHD (2,400x1,600, 235 ppi) LCD touchscreen
Graphics: Intel HD Graphics 615
RAM: 8GB LPDDR3 (1,866MHz)
Storage: 256GB SSD (eMMC)
Ports: 2x USB-C 3.1, headphone/mic jack
Connectivity: 802.11ac Wi-Fi (2x 2 MIMO), Bluetooth 4.2
Cameras: 720p webcam (60fps)
Weight: 1.1kg
Size: 290.4x220.8x10.3mm W x D x H

The Google Pixelbook is, simply put, the best Chromebook ever made: welcome to the Chromebook reimagined. However, getting in on the ground floor of this revolution is going to cost you dearly. The Google Pixelbook is extremely expensive for a Chromebook. Starting at £999 ($999) and capping out at £1,699 ($1,649) – without even counting the £99 ($99) Pixelbook Pen – this is premium hardware with a premium price to match.

For that, you're getting 7th-generation Kaby Lake Intel Core i5 processors on both the entry-level 128GB option and £1,199 ($1,199) mid-range 256GB option, each paired with 8GB of memory. However, the top-end 512GB option comes packing a Core i7 processor and 16GB of memory. All of these processor options are Intel's low-power, low-heat Y series chips, thus all models are fanless.

This is, without a doubt, Google's most attractive and well-conceived computing device yet. From the brushed aluminium frame with flush edges to the rubberised palm rest and underside, every design element has achieved style and substance in equal measure. Well, nearly: Google has crammed the speakers beneath the keyboard, and the result is awfully tinny sound. On the upside, the glass trackpad is a delight to use, tracking super-smoothly and accurately both with single- and-multi-touch gestures. Likewise, the Pixelbook keyboard is among the best we've ever tested. The backlit keyboard's

> The marquee feature of the Pixelbook is its support of Android apps, along with Google Play

keys are well-spaced, and the 0.8mm travel is a delight, with forceful feedback.

With 235 pixels-per-inch (ppi) and accurate colour reproduction, the Pixelbook's display rivals some of the best around, such as the 227-ppi MacBook Pro (13-inch). The panel works well for movies and photos, not to mention photo editing. The 400 nits of brightness help hugely with this, but it's still a glossy screen and as such doesn't stand up to direct sunlight all that well. At any rate, the display is also sharply accurate to the touch, especially when underneath the Pixelbook Pen.

It's a shame that the Pixelbook Pen isn't included as it's arguably crucial to the experience. The pen works excellently as a stylus, offering plenty of pressure response as well as tilt support, making drawing on the display a pleasure.

There's a single button for accessing Google Assistant, but it also incorporates some of the new Google Lens technology found in smartphones such as the Google Pixel 2. Pressing the button while circling something on-screen sends the capture to Google Assistant for analysis. We circled a picture of Ron Livingston in the film Office Space, and Google Assistant spat back his character's name – Peter Gibbons – before telling us more about the actor.

One major flaw of the Pixelbook Pen, though, is that it doesn't attach to the laptop in any way. It also runs on AAAA batteries, when we'd expect a rechargeable solution at this price.

Not surprisingly, the Pixelbook is a strong performer. The laptop can handle entire workloads through the Chrome browser – from Google documents and spreadsheets to chat and photo editing – with nary a hiccup. Google promises up to 10 hours of usage on a single charge. In our battery test, which sees the device loop a 1080p movie at 50% screen brightness and volume, with the keyboard backlight and Bluetooth disabled, the Pixelbook lasted for 7 hours and 40 minutes. That's impressive in its own right, but the cheaper albeit less powerful Asus Chromebook Flip lasted a whopping 10 hours and 46 minutes. Regardless, just 15 minutes on charge gets you up to two hours of use from the Pixelbook, thanks to USB-C fast charging.

The marquee feature of the Pixelbook is its support of Android apps, along with the Google Play store and the brand new launcher interface to access these apps. The result is, frankly, impressive. Every Android app we downloaded, from Sonic the Hedgehog to the VLC video player, worked without issue. Some apps render as if they were on a smartphone, but that's more dependent on the app developers than Google.

Ultimately, this level of Android app support stands to blow Chrome OS wide open, effectively eliminating its dependence on the Chrome web store for app-like experiences. It brings the OS far closer in capability and versatility to full-blown distros.

Joe Osborne

Pros

Sublime design with an excellent keyboard and extra stylus. The first to offer full Android app support

Cons

The stylus is useful but an expensive extra, audio performance is poor and there's no biometric login

Summary

The Google Pixelbook is the first Chromebook that is worthy of your consideration alongside the most high-end devices. We're now at the point where there are little to no compromises for almost anyone to switch to Chromebook from another OS thanks to Android.

10

Above Under the bonnet, many of the Gnome packages have switched to using the Meson build system

DISTRO

Fedora 27

RAM
1GB

Storage
10GB

Specs
1GHz processor

Live installable ISO for 64-bit only, net install for 32-bit

Available from: https://getfedora.org

Is the first Fedora release since Ubuntu's switch to GNOME still the leading GNOME distribution?

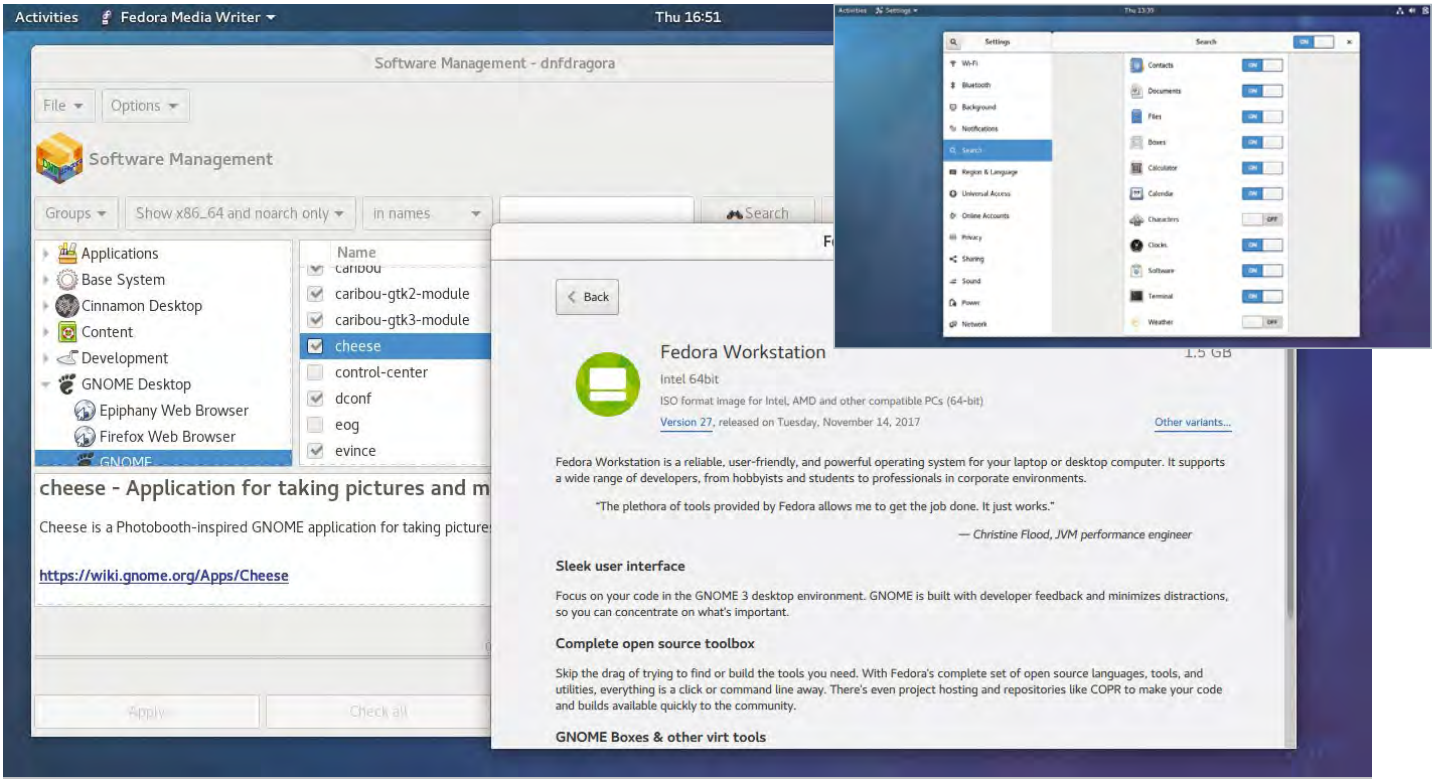
Fedora's well-oiled release machinery has spurted out another update. To keep things lively, however, the project has decided to hop on the bandwagon of mainstream distributions that have trimmed support for the 32-bit platform.

Besides this minor scandalous event, it's pretty much business as usual. The distro ships with a Fedora-branded but largely untouched variant of the stable GNOME 3.26 desktop environment. GNOME's latest offers better Wayland support, some improvements for HiDPI displays, and several minor app improvements.

The developers have also tweaked the Boxes virtualisation app to enable you to easily deploy Red Hat Enterprise Linux 7 virtual machines once you've signed up for a free Red Hat developer account.

There are a slew of visible improvements on the desktop as well, including a redesigned Settings panel with updated panels for both the Display and Network configuration. The new Display settings are really helpful if you have multiple displays and give you a quick overview of how they're set up. You also get buttons to quickly switch between the three supported display modes. The global system search is also now more pervasive and can display system actions such as Suspend and Lock Screen.

On the app front there's the hot-off-the-presses Firefox 57. This version is being hailed as the web browser's biggest release to date, and features a redesigned user interface with a streamlined new core. The other major app is LibreOffice 5.4 which also brings with it new functions and improvements


Activities
Fedora Media Writer
Thu 16:51
Software Management - dnfdragora
File
Options
Software Management
Groups
Show x86_64 and noarch only
in names
Search
Applications
Base System
Cinnamon Desktop
Content
Development
GNOME Desktop
Epiphany Web Browser
Firefox Web Browser
GNOME
Name
caribou-gtk2-module
caribou-gtk3-module
cheese
control-center
dconf
eog
evince
cheese - Application for taking pictures and m
Cheese is a Photobooth-inspired GNOME application for taking picture
https://wiki.gnome.org/Apps/Cheese
Apply
Check all
Back
Fedora Workstation
Intel 64bit
ISO format image for Intel, AMD and other compatible PCs (64-bit)
Version 27, released on Tuesday, November 14, 2017
Other variants...
Fedora Workstation is a reliable, user-friendly, and powerful operating system for your laptop or desktop computer. It supports a wide range of developers, from hobbyists and students to professionals in corporate environments.
"The plethora of tools provided by Fedora allows me to get the job done. It just works."
— Christine Flood, JVM performance engineer
Sleek user interface
Focus on your code in the GNOME 3 desktop environment. GNOME is built with developer feedback and minimizes distractions, so you can concentrate on what's important.
Complete open source toolbox
Skip the drag of trying to find or build the tools you need. With Fedora's complete set of open source languages, tools, and utilities, everything is a click or command line away. There's even project hosting and repositories like COPR to make your code and builds available quickly to the community.
GNOME Boxes & other virt tools

Above Fedora 27 joins another distribution trend of dropping alpha releases in order to free up the release team for other work

> There are a slew of visible improvements on the desktop, including a redesigned Settings panel with updated panels for both Display and Network

in its most popular Writer and Calc apps. The release also gets a major security feature with the ability to use OpenPGP keys to sign ODF documents – and, talking of security, you can now enable trim support for encrypted solid-state drives.

Gnome Software is an aesthetically pleasing app store for new users, but only exposes a small subset of what's available in Fedora's repositories. In earlier releases, advanced users who wanted to avoid the command line switched to Yum Extender, which has since ceased development. It's been replaced with dnfdragora, a new front-end to the DNF package management system written in Python 3 – and as an extra to the default installation, the official Fedora 27 repositories now ship with it.

Another important member of the Fedora family of releases is Atomic Host, which enables you to deploy and host containers. Fedora 27 Atomic Host has switched to a simpler container storage setup that gives you the flexibility to choose different versions of Kubernetes. The other major Fedora variant, Fedora Server, is being reimagined as a more modular server OS. Rather than needing to upgrade the entire server, with a modular server you can have multiple components on different lifecycles. This leads to one other slightly less welcome change: until now, the entire family of Fedora variants have been released together, but due to the nature of its revamp, Fedora Modular Server was still in beta when Fedora 27 was unveiled. It will (hopefully) have been released by the time you read this review.

Besides the Workstation, Atomic and Modular Server editions, there are, as usual, several official spins built around different desktop environments. There's one each for KDE and Cinnamon, as well as for lightweight desktops such as Xfce, MATE-Compiz, LXDE and LXQt. Then there's Fedora's ARM initiative that produces various desktop and server images for ARM-based systems and devices such as the Raspberry Pi.

Existing Fedora users can upgrade to the latest release with a couple of `dnf` commands. The release is also available via the Fedora Media Writer app, which is also useful for creating bootable SD cards for the ARM devices.

Mayank Sharma

Pros

A regular Fedora release made special because of the changes that come with Gnome 3.26 and Firefox 57.

Cons

Ditches 32-bit users, who don't get a Live installable ISO but can use the minimal net install ISO to build from scratch.

Summary

With a barely modified and elegant rendition of the latest Gnome release, Fedora 27 continues to be the marquee Gnome distribution. We can't think of a reason not to upgrade to this release, and it's a perfect time for new users to jump onto the Fedora bandwagon too.

9

ADVANCED TEXT EDITOR

Atom 1.22

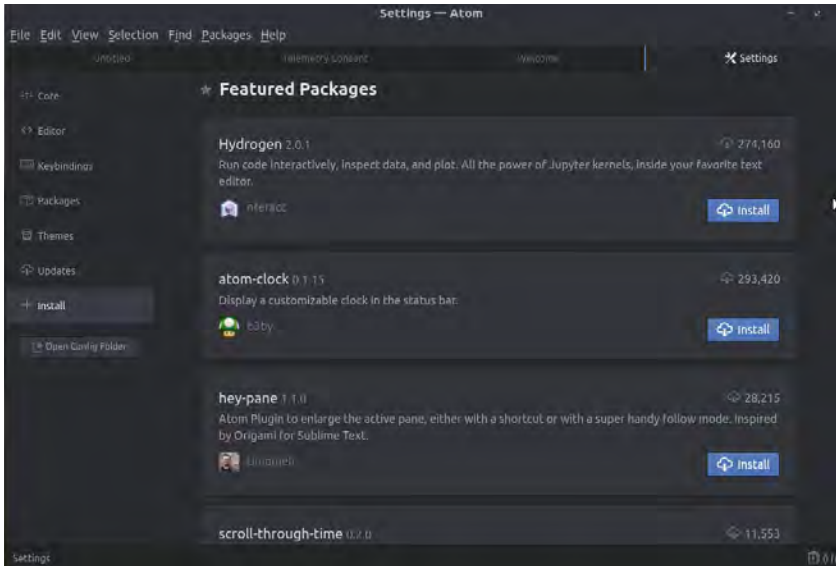
A text editor that can also crunch code

Atom describes itself as a hackable text editor, which isn't an exaggeration considering the amount of customisation that's possible. Developed by GitHub, the app has a built-in package manager that enables you to search and install plug-ins from within the app. About 80 ship with it by default – and there are almost 7,000 more.

One interesting feature is the find-and-replace function which you can use to modify text in a file, or across multiple ones, as you type. Atom can also be used as an IDE and one of its highlights is the smart autocomplete, while the new version introduces a bracket-matching feature that highlights the line number of the closing bracket corresponding to the one under your cursor.

Atom ships with several UI and syntax themes for customisation, and you can even create your own. You can also define custom key bindings and add more functionality with packages for things like minimaps and syntax-specific snippet libraries.

Lastly, since Atom is built on the Electron framework used for creating cross-platform apps using web technologies, it runs on all major OSes.


Settings — Atom
File Edit View Selection Find Packages Help
Settings
Featured Packages
Hydrogen 2.0.1
Run code interactively, inspect data, and plot. All the power of Jupyter kernels, inside your favorite text editor.
Install
atom-clock
Display a customizable clock in the status bar.
293,420
Install
hey-pane
Atom Plugin to enlarge the active pane, either with a shortcut or with a super handy follow mode. Inspired by Origami for Sublime Text.
28,215
Install
scroll-through-time
11,553

Above GitHub has collaborated with Facebook on a set of packages that bring IDE-like functionality to Atom

Pros

A highly customisable app that can be transformed into a very capable IDE with not much effort.

Cons

It's primarily meant for writing code, so transforming it into an advanced text editor takes some doing.

Great for…

Writing and editing code in several languages.
https://atom.io

DESKTOP ENVIRONMENT

Enlightenment 0.22

An aesthetically pleasing desktop that's also lightweight

Enlightenment is a rather unusual desktop environment. In fact, it's more of a window manager, as it lacks taskbars, panels and even menus. Yet, unlike many lightweight environments, the desktop has all the eye-candy you'd expect from a full-blown environment, using a fraction of the resources.

There are subtle animations woven into almost every element of the desktop, from the menus to the various desktop widgets. The desktop has a first-boot wizard that enables you to define various aspects of its behaviour and appearance. For example, there's an option to select text size in windows, which is a really useful feature – and even more so if you're running Enlightenment on a HiDPI display. There's also a unique default behaviour in that windows are selected automatically when you move the cursor over them, although this can be overridden during initial set-up.

Plenty of modules and other configuration options will keep you busy without overwhelming new user; for example, the latest 0.22 release features improved support for the Wayland display server. You also get a new graphical sudo/ssh askpass utility, as well as volume controls integrated in all media-playing windows.

The best way to install Enlightenment is through your distribution's repositories. Ubuntu users can get it by adding its PPA using `sudo add-apt-repository ppa:niko2040/e19`

Pros

A desktop that's loaded with bling, but still very conscious of its resource usage.

Cons

The lack of traditional desktop furniture can take some getting used to.

Great for…

An attractive looking desktop on a low-end machine.
www.enlightenment.org

PROGRAMMING LIBRARY

OpenCV 3.3.1

Program your computer to recognise and track all kinds of things

The Open Source Computer Vision (OpenCV) is a machine-learning library of programming functions designed for aiding the development of real-time computer vision. The library can read images and detect shapes (circle, square and so on) as well as objects (coins, houses and the like). The functions are also capable of picking out and identifying text in images, such as reading number plates or street signs, which makes it ideal for developing augmented reality apps.

It can recognise faces, gestures and motion too, meaning it's useful for all kinds of robotics applications. In fact, OpenCV is the primary vision library in the popular robotics middleware, Robot Operating System (ROS).

OpenCV was originally developed by Intel Research in 1999 and is now maintained by the OpenCV.org non-profit foundation. The BSD-licensed library is easy to learn, with loads of documentation on its website and elsewhere on the internet, and support for languages such as C++, Java and Python.

Many of the changes in this 3.3.1 release were developed as part of the Google Summer of Code 2017, including end-to-end text detection and recognition. Another major change are improvements to the library's Deep Neural Network module, with the addition of several new samples. In addition to Linux, OpenCV runs on all major desktop and mobile OSes; the installation process is fairly straightforward but quite involved, and it's thoroughly documented on the website.

Pros

Allows coders of popular languages to add image-recognition capabilities to their projects.

Cons

There's an involved installation process, and a learning curve, as with any programming module.

Great for…

Infusing artificial intelligence via vision in computers.

https://opencv.org

WEB BROWSER

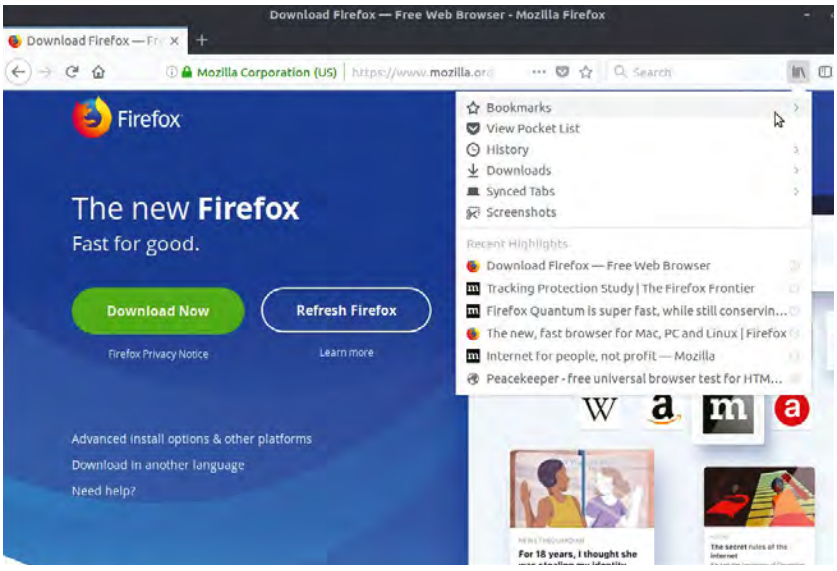
Firefox 57

A watershed release, this version is a must-try

Mozilla is quite excited about its latest Firefox release, codenamed Quantum. It claims the new version is twice as fast as previous releases according to the Speedometer 2.0 benchmark. These performance improvements are a result of better multi-threading, a new CSS engine rewritten in Rust, and other components borrowed from the Servo layout engine. While your results may vary, you'll almost certainly notice an improvement in launch times and page loading, more responsive tab-switching and smoother scrolling.

The browser also features a redesigned user interface, and of particular note is the new Firefox Library menu in the redesigned toolbar. The Library gathers your browsing history, bookmarks, synced tabs, downloads and more into one area. The New Tab page has also been revamped, and now includes top and recently visited sites and pages, along with recommendations from Pocket for users in Canada, the US and Germany.

You can wait for the new version to make its way to your distribution's official repositories, or just grab the compressed archive from the Firefox website.


Download Firefox — Free Web Browser - Mozilla Firefox
Firefox
The new Firefox
Fast for good.
Download Now
Refresh Firefox
Firefox Privacy Notice
Learn more
Advanced install options & other platforms
Download in another language
Need help?
Bookmarks
View Pocket List
History
Downloads
Synced Tabs
Screenshots
Recent Highlights
Download Firefox — Free Web Browser
Tracking Protection Study | The Firefox Frontier
Firefox Quantum is super fast, while still conservin...
The new, fast browser for Mac, PC and Linux | Firefox
Internet for people, not profit — Mozilla
Peacekeeper - free universal browser test for HTM...

Above The tracking protection in the Private Browsing mode blocks certain scripts, for faster page loading

Pros

Incorporates major changes to the UI and the core code that improve performance across the board.

Cons

Pocket integration is limited, and changes to the Add-ons format might render some extensions defunct.

Great for…

Making better use of resources, especially on slower machines.

www.mozilla.org

Get your listing in our directory
To advertise here, contact Chris
chris.mitchell@futurenet.com | +44 01225 68 7832 (ext. 7832)

RECOMMENDED

Hosting listings

Featured host:

www.thenames.co.uk
0370 321 2027

TheNames

About us

Part of a hosting brand started in 1999, we're well-established, UK-based, independent and our mission is simple – ensure your web presence 'just works'.

We offer great-value domain names, cPanel web hosting, SSL certificates, business email, WordPress hosting, cloud and VPS.

What we offer

- Free email accounts with fraud, spam and virus protection
- Free DNS management
- Easy-to-use Control Panel
- Free email forwards – automatically redirect your email to existing accounts
- Domain theft protection to prevent it being transferred out accidentally or without your permission
- Easy-to-use bulk tools to help you register, renew, transfer and make other changes to several domain names in a single step
- Free domain forwarding to point your domain name to another website

5 Tips from the pros

01 **Optimise your website images**
When uploading your website to the internet, make sure all of your images are optimised for the web. Try using **jpegmini.com** software; or if using WordPress, install the EWWW Image Optimizer plugin.

02 **Host your website in the UK**
Make sure your website is hosted in the UK, and not just for legal reasons. If your server is located overseas, you may be missing out on search engine rankings on google.co.uk – you can check where your site is based on **www.check-host.net**.

03 **Do you make regular backups?**
How would it affect your business if you lost your website today? It's vital to always make your own backups; even if your host offers you a backup solution, it's important to take responsibility for your own data and protect it.

04 **Trying to rank on Google?**
Google made some changes in 2015. If you're struggling to rank on Google, make sure that your website is mobile-responsive. Plus, Google now prefers secure (HTTPS) websites. Contact your host to set up and force HTTPS on your website.

05 **Avoid cheap hosting**
We're sure you've seen those TV adverts for domain and hosting for £1! Think about the logic... for £1, how many clients will be jam-packed onto that server? Surely they would use cheap £20 drives rather than £1k+ enterprise SSDs? Remember: you do get what you pay for.

Testimonials

David Brewer
"I bought an SSL certificate. Purchasing is painless, and only takes a few minutes. My difficulty is installing the certificate, which is something I can never do. However, I simply raise a trouble ticket and the support team are quickly on the case. Within ten minutes I hear from the certificate signing authority, and approve. The support team then installed the certificate for me."

Tracy Hops
"We have several servers from TheNames and the network connectivity is top-notch – great uptime and speed is never an issue. Tech support is knowledge and quick in replying – which is a bonus. We would highly recommend TheNames. "

J Edwards
"After trying out lots of other hosting companies, you seem to have the best customer service by a long way, and all the features I need. Shared hosting is very fast, and the control panel is comprehensive..."

Supreme hosting

CWCS MANAGED HOSTING — SUPREME HOSTING. SUPREME SUPPORT.

www.cwcs.co.uk
0800 1 777 000

CWCS Managed Hosting is the UK's leading hosting specialist. It offers a fully comprehensive range of hosting products, services and support. Its highly trained staff are not only hosting experts, it's also committed to delivering a great customer experience and is passionate about what it does.

- Colocation hosting
- VPS
- 100% Network uptime

Value hosting

elastichosts

elastichosts.co.uk
02071 838250

ElasticHosts offers simple, flexible and cost-effective cloud services with high performance, availability and scalability for businesses worldwide. Its team of engineers provide excellent support around the clock over the phone, email and ticketing system.

- Cloud servers on any OS
- Linux OS containers
- World-class 24/7 support

Small business host

HOSTPAPA

www.hostpapa.co.uk
0800 051 7126

HostPapa is an award-winning web hosting service and a leader in green hosting. It offers one of the most fully featured hosting packages on the market, along with 24/7 customer support, learning resources and outstanding reliability.

- Website builder
- Budget prices
- Unlimited databases

Enterprise hosting:

2020Media www.2020media.com

www.2020media.com | 0800 035 6364

WordPress comes pre-installed for new users or with free managed migration. The managed WordPress service is completely free for the first year.

We are known for our "Knowledgeable and excellent service" and we serve agencies, designers, developers and small businesses across the UK.

Budget hosting:

HETZNER ONLINE

www.hetzner.de/us | +49 (0)9831 5050

Hetzner Online is a professional web hosting provider and experienced data-centre operator. Since 1997 the company has provided private and business clients with high-performance hosting products, as well as the necessary infrastructure for the efficient operation of websites. A combination of stable technology, attractive pricing and flexible support and services has enabled Hetzner Online to continuously strengthen its market position both nationally and internationally.

- Dedicated and shared hosting
- Colocation racks
- Internet domains and SSL certificates
- Storage boxes

SSD web hosting

bargainhost

www.bargainhost.co.uk
0843 289 2681

Since 2001, Bargain Host has campaigned to offer the lowest-priced possible hosting in the UK. It has achieved this goal successfully and built up a large client database which includes many repeat customers. It has also won several awards for providing an outstanding hosting service.

- Shared hosting
- Cloud servers
- Domain names

Value Linux hosting

PATCHMAN WEB HOSTING

patchman-hosting.co.uk
01642 424 237

Linux hosting is a great solution for home users, business users and web designers looking for cost-effective and powerful hosting. Whether you are building a single-page portfolio, or you are running a database-driven ecommerce website, there is a Linux hosting solution for you.

- Student hosting deals
- Site designer
- Domain names

Fast, reliable hosting

:BYTEMARK

www.bytemark.co.uk
01904 890 890

Founded in 2002, Bytemark are "the UK experts in cloud & dedicated hosting". Its manifesto includes in-house expertise, transparent pricing, free software support, keeping promises made by support staff and top-quality hosting hardware at fair prices.

- Managed hosting
- UK cloud hosting
- Linux hosting

Free Gifts

Get your free resources

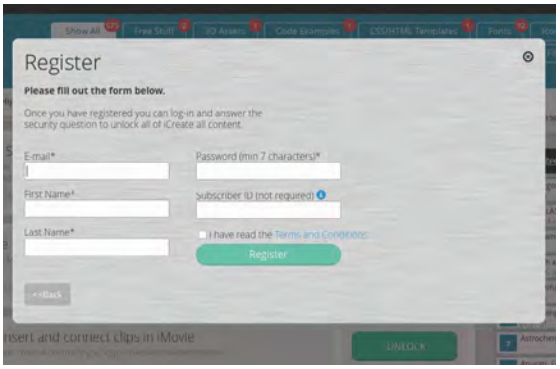
Download the best distros, essential FOSS and all our tutorial project files from your FileSilo account

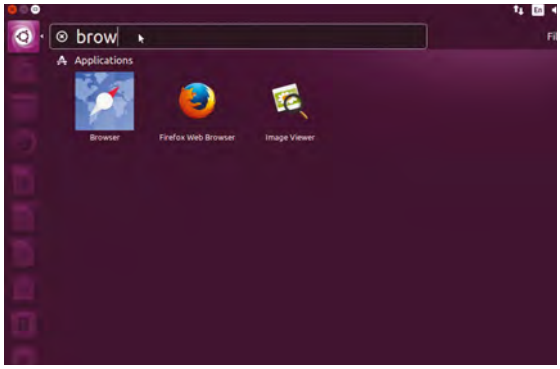
WHAT IS IT?

Every time you see this symbol in the magazine, there is free online content that's waiting to be unlocked on FileSilo.

WHY REGISTER?

- Secure and safe online access, from anywhere
- Free access for every reader, print and digital
- Download only the files you want, when you want
- All your gifts, from all your issues, all in one place


Register
Please fill out the form below.
Once you have registered you can log-in and answer the security question to unlock all of iCreate all content.
E-mail*
Password (min 7 characters)*
First Name*
Subscriber ID (not required)
Last Name*
I have read the Terms and Conditions
Register



1. UNLOCK YOUR CONTENT

Go to **www.filesilo.co.uk/linuxuser** and follow the instructions on screen to create an account with our secure FileSilo system. When your issue arrives or you download your digital edition, log into your account and unlock individual issues by answering a simple question based on the pages of the magazine for instant access to the extras. Simple!

2. ENJOY THE RESOURCES

You can access FileSilo on any computer, tablet or smartphone device using any popular browser. However, we recommend that you use a computer to download content, as you may not be able to download files to other devices. If you have any problems with accessing content on FileSilo, take a look at the FAQs online or email our team at **filesilohelp@futurenet.com**.

Free for digital readers too!
Read on your tablet, download on your computer

Log in to **www.filesilo.co.uk/linuxuser**

DISTRO
Diet Pi v150

DISTRO
Raspbian Stretch Lite - November 2017

DISTRO
Zorin OS 12.2

DISTRO
Linux Mint 18.3 Cinnamon

Subscribe and get instant access

Get access to our entire library of resources with a money-saving subscription to the magazine – subscribe today!

This month find...

DISTROS

Two established distros this month: Linux Mint 18.3 Cinnamon, with its revamped Software Manager, plus Zorin OS 12.2 Core, which also includes Wine 2.0 for better support of Windows apps.

SOFTWARE

To accompany our roundup of Raspberry Pi desktop OSes, we've got two of the best. Try Raspbian Stretch Lite and DietPi v150 for a streamlined approach.

TUTORIAL CODE

Complete code for every tutorial, including the skeleton project for MQTT, Python code for Arduino, a TAR for the Java series and lots more!

FREE DVD! 4.7GB OF DISTROS & TOOLS
Linux User & Developer
>_100 WAYS TO
MASTER THE TERMINAL
Become a command line guru in 24 hours
49 EXPERT GUIDES

Subscribe & save!

See all the details on how to subscribe on **page 30**

FOLLOW US
Facebook: facebook.com/LinuxUserUK
Twitter: @linuxusermag

NEAR-FUTURE FICTION

Deliver me from darkness

Where was that bloody delivery drone?

He'd been waiting for three hours, from the moment he'd woken up.

How many times would he have to stay at home on the promise that his new eyes would be arriving that day?

Okay, so he'd not chosen guaranteed next-day delivery, but at the time he'd ordered them his eyes still had a good four weeks left in them. And yes, he'd been a bit casual about making sure he was there to sign for them, but the more critical it was getting the less the company seemed to want to help.

They insisted a drone had been at his door every day, but he'd been there most days. It was a load of rubbish. They just didn't care.

The light faded a little. His eyes were on their last legs, so to speak.

If he didn't get his new eyes soon then his vision would cease, and no matter how many replacements they delivered he wouldn't be able to see to install them. It was a dire situation.

He swiped his phone to check the delivery.

He couldn't quite make out what it said.

Why the bloody hell couldn't he just call them like in the old days? Ring and speak to someone, or at least have an online chat.

The room got darker and in the corner of his eye he could see the energy level was down to the last notch of the last bar.

How long did a notch last?

He couldn't remember.

The doorbell rang.

At last!

His sight faded to nothing but he managed to stumble across the room.

'About bloody time,' he said as he yanked open the door.

'Hi,' said his neighbour, 'I seem to have a parcel that was meant for you.'

Tears welled up and flooded his face. 'I can't see... My eyes have failed... Are these the new ones?'

'Hold on. I'll check,' said his neighbour. 'Yes, they are.'

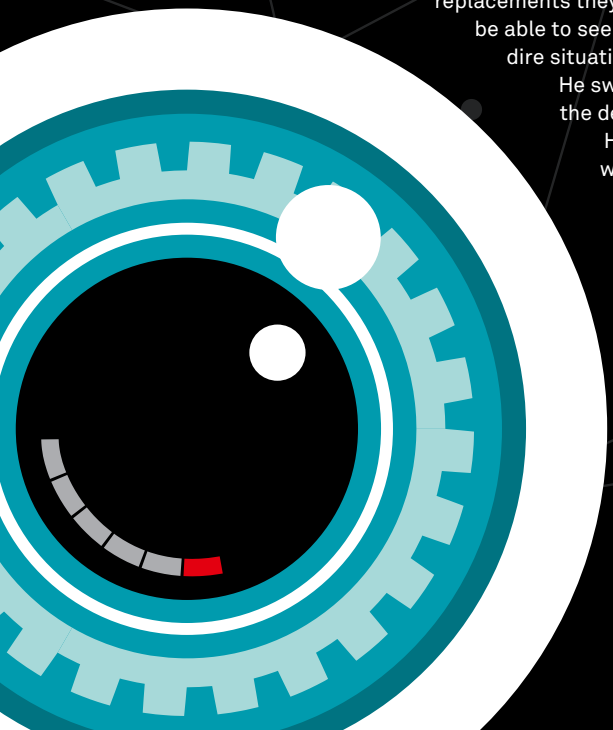
'Too late.' He sobbed.

'Would you like me to install them for you?'

'Oh, yes please. Yes please.'

Overcome by gratitude and relief, he steadied himself on the door.

'Yes please,' he repeated again and again. ■



ABOUT

Eating Robots

Taken from the new book *Eating Robots* by Stephen Oram: near-future science-fiction exploring the collision of utopian dreams and twisted realities as humanity and technology become ever more intertwined. Sometimes funny and often unsettling, these 30 sci-fi shorts will stay with you long after you've turned the final page.
http://stephenoram.net

NEXT ISSUE **ON SALE 11 JANUARY**
Virtualise Your System | Protect Your Tech

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier

LOSE YOURSELF IN A WORLD OF
Vinyl
FIND YOURSELF IN
OXFAM'S ONLINE SHOP
oxfam.org.uk/shop
TIME ON MY HANDS
COLUMBIA
OXFAM